

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CENTRO DE INFORMACION Y DOCUMENTACION
"ING. BRUNO MASCANZONI"

EL CENTRO DE INFORMACION Y DOCUMENTACION "ING. BRUNO MASCANZONI" TIENE POR OBJETIVO SATISFACER LAS NECESIDADES DE ACTUALIZACION AL PROPORCIONAR LA ADECUADA INFORMACION QUE PERMITA A LOS PROFESIONALES INGENIEROS PROFESORES Y ALUMNOS, ESTAR AL TANTO DEL ESTADO ACTUAL DEL CONOCIMIENTO SOBRE TEMAS ESPECIFICOS ENFATIZANDO LAS INVESTIGACIONES DE VANGUARDIA DE LOS CAMPOS DE LA INGENIERIA TANTO NACIONALES COMO EXTRANJERAS.

POR LO QUE SE PONE A DISPOSICION DE LOS ASISTENTES DE LOS CURSOS DE LA D.E.C.F.I.; ASI COMO AL PUBLICO EN GENERAL.

EN DICHO CENTRO USTED TENDRA LOS SIGUIENTES SERVICOS:

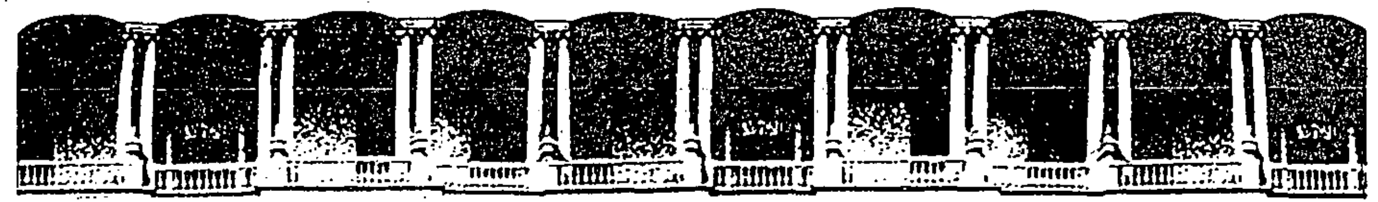
* PRESTAMO INTERNO
* PRESTAMO EXTERNO
* PRESTAMO INTERBIBLIOTECARIO
* SERVICIO DE FOTOCOPIADO
* CONSULTA TELEFONICA
* CONSULTA A LOS BANCOS DE DATOS: LIBRUNAM EN CD-ROM Y EN LINEA

LOS MATERIALES A SU DISPOSICION SON:

* LIBROS
* TESIS DE POSGRADO
* NOTICIAS TECNICAS
* PUBLICACIONES PERIODICAS
* PUBLICACIONES DE LA ACADEMIA MEXICANA DE INGENIERIA
* NOTAS DE LOS CURSOS QUE SE HAN IMPARTIDO DE 1971 A LA FECHA

EN LAS AREAS DE INGENIERIA INDUSTRIAL, CIVIL, ELECTRONICA, CIENCIAS DE LA TIERRA, MECANICA Y ELECTRICA Y COMPUTACION.

EL C.I.D. SE ENCUENTRA UBICADO EN EL MEZZANINE DEL PALACIO DE MINERIA LADO ORIENTE. EN HORARIO DE SERVICIO DE 10:00 A 19:30 HORAS DE LUNES A VIERNES.



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS DE LA DIVISION DE EDUCACION CONTINUA

Las autoridades de la Facultad de Ingeniería, por conducto del Jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia sólamente a los alumnos que tengan un mínimo del 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tésis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con ésto sean más fehacientes sus apreciaciones.

¡ G R A C I A S !

Uno de los proyectos que actualmente esta llevando a cabo la DECFI, es la organización de cursos de actualización en temas de ingeniería, dentro de los cuales se incluyen - programas de computadora relacionados con el tema del curso, los cuales se distribuirán en sus versiones fuente.

Con el objeto de conocer los temas de mayor interés para este tipo de cursos, así como para definir los requisitos técnicos que deben reunir los programas a distribuir, mucho agradeceremos a usted se sirva llenar el siguiente cuestionario, el cual será de - una gran ayuda para la DECFI.

1.- Califique con escala de cero a diez los siguientes cursos utilizando las líneas en blanco para aquellos que usted proponga (0=no interesa, 10 = interesa mucho)

Análisis Estructural	( )	Estadística	( )	Control de Personal	( )
Control de Obras	( )	Diseño Mecánico	( )	Almacenes	( )
Ruta Crítica	( )	Programación Estruc.	( )	Inv. de Operaciones	( )
Programación Lineal	( )	Estructura de Datos	( )	Control de Calidad	( )
Matemáticas	( )	Contabilidad	( )	Admon. Programación de la Producción	( )
____________	( )	____________	( )	____________	( )
____________	( )	____________	( )	____________	( )
____________	( )	____________	( )	____________	( )

Debido a que la principal característica de los cursos sería la de distribuir programas de computadora que puedan ser usadas por los asistentes en sus diferentes empresas con el menor esfuerzo de adaptación.

2.- ¿Para que tipo de computadora desearía que se escribieran los programas?

Primera Opción Marca ____________ Modelo ____________ Lenguaje ____________

Segunda Opción Marca ____________ Modelo ____________ Lenguaje ____________

Tercera Opción Marca ____________ Modelo ____________ Lenguaje ____________

Si usted conoce algunas otras personas interesadas en este tipo de cursos, mucho le - agradeceremos hacerle llegar una copia de esta hoja y enviarla posteriormente a:

División de Educación Continua
Palacio de Minería
Calle de Tacuba No. 5
Delegación Cuauhtemoc
06000 México, D.F.



CUESTIONARIO PARA EL DIRECTORIO GENERAL

PARA USO EXCLUSIVO DE LA D.E.C.

ALTA [SI] MODIFICACION [SI]

1.— INSTRUCCIONES GENERALES.

A.— Escriba con letra de molde.

B.— Escriba un solo caracter por cuadro ó semicuadro.

C.— Para contestar las preguntas en las líneas ó en los semicuadros escriba, letras y en los cuadros pondrá números según sea la respuesta que se pide.

NOTA : No se haga ninguna anotación en los cuadros sombreados.

NOMBRE DEL CURSO : ______________________ FOLIO : [ ][ ][ ][ ][ ] DIG. VER. [ ] 1 5 6

1.— NOMBRE :

* 8 TITULO 13 (PROFESIONAL)

14 NOMBRE (S) APELLIDO PATERNO APELLIDO MATERNO. 41

NOTA : Separe mediante una coma (,) el (los) nombre (s) de los apellidos.

2.— REGISTRO FEDERAL DE CAUS. : 42 AÑO MES DIA 51

3.— CEDULA PROFESIONAL: 52 NUMERO 58

4.— TELEFONO PARTICULAR: 59 NUMERO 65

5.— TELEFONO OFICINA: 66 NUMERO 72 73 EXT. 76

6.— ASISTENTE AL CURSO O PROFESOR: ______________________ 77 A [1] 80 B [5] 80

7.— DIRECCION PARTICULAR :

8 CALLE, NUMERO EXTERIOR E INTERIOR 41

42 COLONIA 71 A [2] 80 M [6] 80

8 DELEGACION O CIUDAD 27 ______________ ESTADO 28 29

CODIGO POSTAL 30 34

8.— ______________________ TITULO PROFESIONAL Y AREA (EJEMPLO: INGENIERO CIVIL) 35 36

______________________ ESPECIALIDAD 37 38

10.– DIRECCION DE OFICINA:

39	CALLE, NUMERO EXTERIOR E INTERIOR	72	A **3** (80) M **7** (80)
8	COLONIA	37	
38	DELEGACION O CIUDAD	57	ESTADO 58 59

CODIGO POSTAL 60 64

11.– ASOCIACIONES A LAS QUE PERTENECE :

PRINCIPAL : ______ 65 66

OTRAS : ______ 67 68

______ 69 70

______ 71 72

______ 73 74 A **4** (80) M **8** (80)

FECHA DE ELABORACION :

______ A ______ DE ______ DE 19 ______

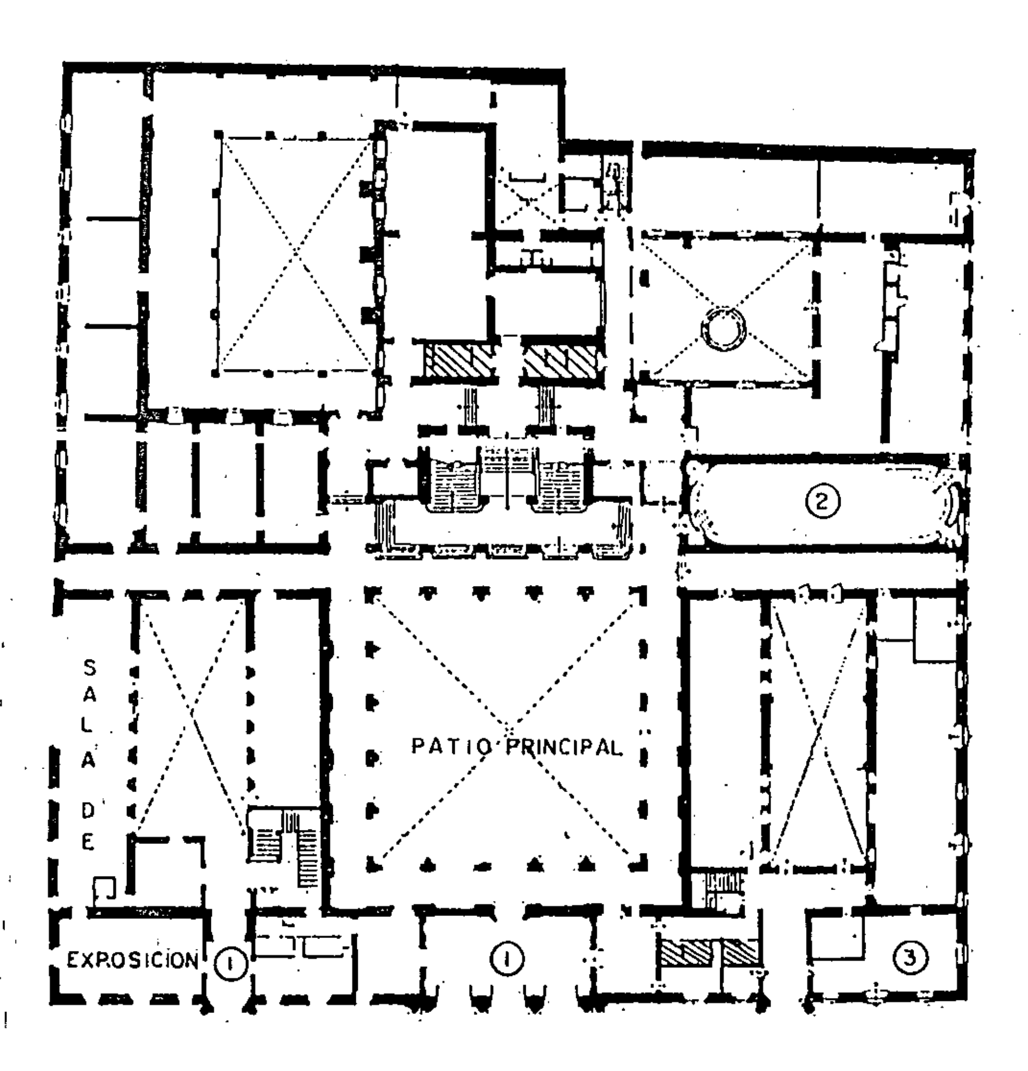
______
FIRMA

PARA USO EXCLUSIVO DE LA DIVISION DE EDUCACION CONTINUA

CODIFICO:	REVISO:	OBSERVACIONES:

DFCEC-01/B

PALACIO DE MINERIA



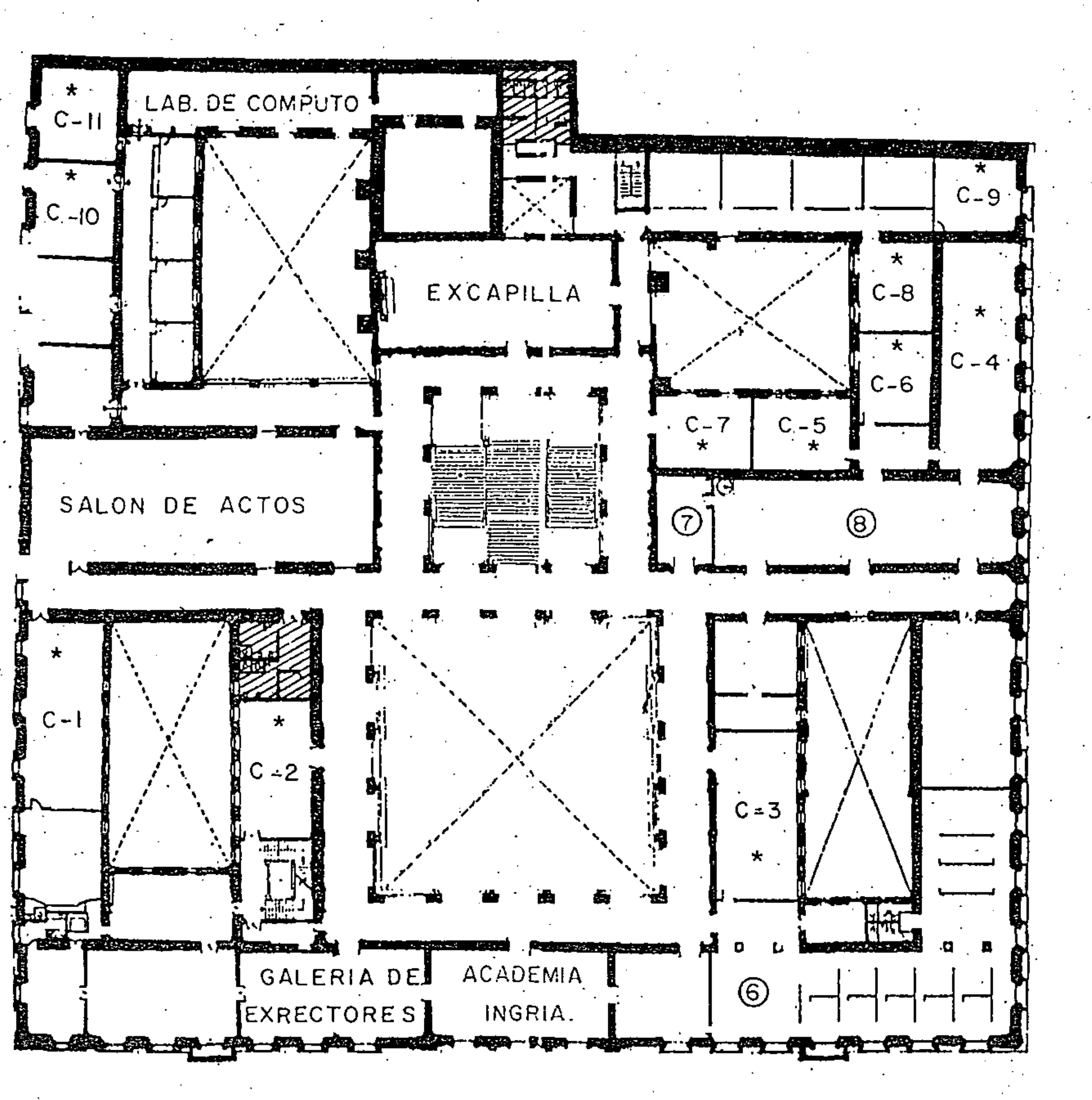


B-7
B-6
B-5
B-4
ANEXO AUDITORIO
AUDITORIO B.Q.
4
5

MEZZANINNE



DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.
CURSOS ABIERTOS



1er. PISO

GUIA DE LOCALIZACION

1 – ACCESO

2 – BIBLIOTECA HISTORICA

3 – LIBRERIA U N A M

4 – CENTRO DE INFORMACION Y DOCUMENTACION "ING. BRUNO MASCANZONI"

5 – PROGRAMA DE APOYO A LA TITULACION

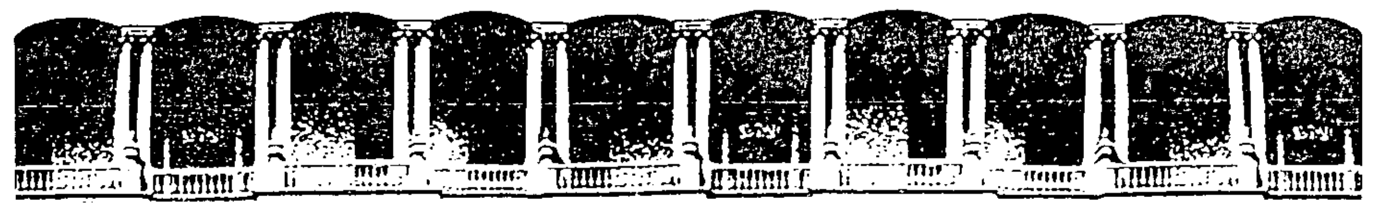
* AULAS

6 – OFICINAS GENERALES

7 – ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA.

8 – SALA DE DESCANSO

▧ SANITARIOS



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

INTRODUCCION A LA PROGRAMACION DE GRAFICAS PARA COMPUTADORAS PERSONALES

M A T E R I A L   D I D A C T I C O

JUNIO-JULIO ,1992.

Palacio de Minería   Calle de Tacuba 5   Primer piso   Deleg. Cuauhtémoc 06000   México, D.F.   Tel.: 521-40-20   Apdo. Postal M-2285

INTRODUCCION A LA PROGRAMACION DE GRAFICAS
PARA COMPUTADORAS PERSONALES

1 CONCEPTOS BASICOS DE GRAFICACION

Esta sección contiene una amplia descripción de los conceptos generales relacionados con el mostrado de gráficas, los cuales son necesarios para comprender el funcionamiento de todas las operaciones de este tipo.

Toda computadora "PC-compatible" cuenta con una parte del equipo, llamada "tarjeta controladora de video" o simplemente tarjeta de video, que está encargada de controlar la presentación de todo lo que aparece en la pantalla de la computadora, como pueden ser textos, dibujos, colores, etcétera.

Las tarjetas de video han evolucionado junto con las computadoras contándose actualmente con diversos modelos que ofrecen múltiples características de operación. Estas tarjetas se conocen por las siglas de sus nombres, por la marca de la compañía que las fabrica (cuando no es IBM) o por su número de parte. A continuación se muestran las tarjetas de video estándar más comunes.

MDA	Monochrome Display Adapter	HERCULES	(Marca de compañía)
CGA	Color Graphics Adapter	ATT-400	(Marca de compañía)
MCGA	Multi Color Graphics Array	PC-3270	(Número de parte)
EGA	Enhanced Graphics Adapter	IBM-8514	(Número de parte)
VGA	Video Graphics Array		

Además de las anteriores también existen algunas tarjetas de video que pueden emular la operación de dos o más de las tarjetas estándar, por lo que tales tarjetas cuentan con la ventaja de poder operar como una u otra tarjeta de video dependiendo de los requerimientos de un programa.

Las tarjetas de video más sencillas, como la MDA, solamente pueden mostrar texto en la pantalla, pero las más avanzadas también pueden mostrar gráficas mediante la técnica de dividir la pantalla en un cierto número de puntos, llamados "pixels", los cuales se encienden o apagan en forma individual. Una tarjeta de video que puede mostrar gráficas se llama "tarjeta de gráficas".

La cantidad de pixels en que se divide la pantalla se llama "resolución" y se indica por medio de dos números que marcan la cantidad de pixels en sentido horizontal y vertical, respectivamente. Por ejemplo, si se divide la pantalla horizontalmente en 320 pixels y verticalmente en 200 pixels, se dice que tenemos una resolución de 320 x 200. Para identificar un pixel determinado dentro de la pantalla se emplea un par de números, llamados "coordenadas", que indican la posición horizontal y vertical, respectivamente, del pixel deseado.

Las coordenadas comienzan en la esquina superior izquierda de la pantalla y van aumentando hacia la derecha y hacia abajo, pero es importante recordar que estas posiciones comienzan en cero, no en uno. De esta manera, las coordenadas (0,0) siempre indicarán la esquina superior izquierda de la pantalla, y si se tiene una resolución de 320 x 200 pixels la esquina superior derecha estará en (319,0), la inferior izquierda en (0,199) y la inferior derecha en (319,199).

Para trazar figuras y líneas en la pantalla se deben encender una serie de pixels adyacentes de tal forma que todos juntos formen la línea deseada. Las líneas verticales y horizontales siempre aparecerán correctamente en la

pantalla, pero las líneas inclinadas estarán formadas por una serie de segmentos de líneas horizontales, verticales o pixels solos que parecerán "escalones" dándole a la línea inclinada el aspecto de una pequeña escalera.

Este efecto será mayor cuanto más grandes sean los pixels individuales que forman la línea, por lo que es conveniente que los pixels sean lo más pequeño posible lo que se consigue con una resolución alta, ya que mientras más pixels haya en el mismo espacio éstos serán más pequeños. Cada nuevo tipo de tarjeta de video que sale al mercado generalmente ofrece una mayor resolución que los modelos anteriores; por ejemplo, la tarjeta CGA tiene una resolución máxima de 640 x 200 pixels, la EGA de 640 x 350, la VGA de 640 x 480, etcétera.

Las líneas, figuras y zonas iluminadas de la pantalla se forman con grupos de pixels que se encienden juntos; por lo tanto, el proceso de mostrar figuras determinadas en posiciones determinadas, como pueden ser líneas, cuadrados, polígonos, círculos, elipses, etc., consiste en calcular las coordenadas de todos los pixels que forman la figura y encenderlos. Hacer este tipo de cálculos es muy laborioso y puede ser muy complicado, por lo que normalmente los lenguajes de programación cuentan con una serie de herramientas de graficación que permiten realizar estas operaciones en una forma muy sencilla.

Dependiendo del tipo de la tarjeta de video los pixeles de la pantalla se podrán encender en diferentes colores, lo que permite mostrar un cierto número de colores al mismo tiempo. De esta forma se pueden tener desde pixels de un sólo color sobre fondo negro en la tarjeta HERCULES (es decir, 2 colores simultáneos), 3 colores sobre un fondo de otro color en la tarjeta CGA (4 colores simultáneos), 16 colores simultáneos en las tarjetas EGA y VGA hasta llegar a 256 colores simultáneos en la nueva tarjeta IBM-8514.

El color que tiene cada pixel se identifica con un número que empieza en 0 y que puede llegar hasta el máximo número de color permitido por la tarjeta de video, el cual será un 1, 3, 15 ó 255 cuando se tengan 2, 4, 16 ó 256 colores simultáneos, respectivamente. El número 0 siempre corresponde al fondo de la pantalla, es decir, todos los pixels que estén apagados tendrán el color 0 mientras que los demás números de color indicarán otros colores distintos.

Si dos figuras se trazan con los colores números 1 y 2 ambas aparecerán con colores diferentes, pero el color real con que se muestren en la pantalla (rojo, verde, etc.) se podrá cambiar dentro de ciertos límites dependiendo del tipo de tarjeta de video. Por ejemplo, el color número 0 siempre indicará el color del fondo de la pantalla, pero éste podrá ser negro, azul, gris, etc. dependiendo del "matiz" que se haya asignado previamente al color número 0.

Las tarjetas EGA y VGA cuentan con 64 matices diferentes, lo cual quiere decir que pueden mostrar 16 colores al mismo tiempo escogiéndolos entre una "paleta de colores" que tiene 64 posibilidades. En este caso los matices son números que varían entre 0 y 63, los cuales cubren la gama de matices desde el negro hasta el blanco, respectivamente. Cuando estas tarjetas se activan por primera vez sus 16 colores se inicializan con ciertos matices estándar, los cuales se muestran a continuación: 0=negro, 1=azul, 2=verde, 3=cyan (azul-verde), 4=rojo, 5=morado, 6=café, 7=gris claro, 8=gris oscuro, 9=azul claro, 10=verde claro, 11=cyan claro, 12=rojo claro, 13=morado claro, 14=amarillo y 15=blanco.

Si se muestran en la pantalla una serie de figuras con zonas de diversos colores y luego se cambian repetidamente los matices asignados a cada uno de sus números de color, se pueden conseguir interesantes efectos que simulan movimientos en una forma muy realista. Estos efectos serán más convincentes mientras mayor sea el número de colores simultáneos y matices que se puedan cambiar. Las tarjetas de video que pueden mostrar 256 colores simultáneos nos permiten escogerlos entre la asombrosa cantidad de 262,144 matices distintos.

Para que una tarjeta de gráficas pueda mostrar texto o gráficas según se requiera es necesario cambiar la forma de presentación de los datos en la pantalla, ya que ambas presentaciones requieren un manejo interno diferente. Para lograr esto, las tarjetas de gráficas permiten varias formas distintas de operación, llamadas "modos de video", los cuales proporcionan diferentes características dependiendo del modo de video en sí que se seleccione.

Por ejemplo, la tarjeta CGA permite dos modos de video para presentación de gráficas: el primero tiene una resolución "baja" de 320 x 200 pixels con 4 colores simultáneos, mientras que el segundo es la resolución "alta" de 640 x 200 pixels pero sólo permite un color sobre fondo negro. Por lo tanto, si se selecciona el primer modo se tendrán 4 colores con líneas gruesas y eligiendo el segundo habrá líneas delgadas pero solamente de un color, por lo que será necesario elegir el modo de video adecuado en base a los requerimientos de graficación que se tengan (colores deseados contra definición de líneas).

En forma similar, las demás tarjetas de video cuentan con diversos modos de video gráfico que pueden o no tener características semejantes. En particular las tarjetas de video CGA, EGA, VGA e IBM-8514 presentan una evolución y compatibilidad continuas, lo cual quiere decir que cada nueva tarjeta de esta serie puede operar en los mismos modos de video que las tarjetas anteriores y adicionalmente en algunos modos nuevos con mayores capacidades. Los distintos modos de video se indican mediante un número que varía desde 0 hasta el último modo de video disponible de acuerdo con el tipo de tarjeta, correspondiendo el número más alto al modo de video con mayor resolución; cada uno de estos modos tendrá una cierta resolución y un número de colores simultáneos.

Para terminar con esta descripción se muestra una breve tabla con las características generales de las tarjetas de video mas difundidas y un resumen de los conceptos generales involucrados en el manejo de gráficas.

CARACTERISTICAS DE LAS TARJETAS DE VIDEO ESTANDAR MAS COMUNES

MDA (Monochrome Display Adapter)
No permite gráficas, sólo muestra texto a un sólo color (el de la pantalla, verde o ámbar) sobre negro.

HERCULES (marca de compañía)
Tiene un sólo modo de gráficas con una resolución de 720 x 348 a un sólo color (el de la pantalla, verde o ámbar) sobre negro.

CGA (Color Graphics Adapter)
Tiene dos modos de gráficas: la resolución "baja" de 320 x 200 a 4 colores simultáneos, y la resolución "alta" de 640 x 200 a 1 color sobre negro.

EGA (Enhanced Graphics Adapter)
Tiene dos modos nuevos de gráficas con resoluciones de 640 x 200 y de 640 x 350, ambos a 16 colores simultáneos de una paleta de 64 posibles.

VGA (Video Graphics Array)
Tiene un modo nuevo de gráficas con una resolución de 640 x 480 a 16 colores simultáneos de una paleta de 64 posibles.

IBM-8514 (número de parte)
Tiene dos modos nuevos de gráficas con resoluciones de 640 x 480 y de 1024 x 768, ambos a 256 colores simultáneos de una paleta de 262,144 posibles.

RESUMEN DE CONCEPTOS GRAFICOS BASICOS

1.- Las características de las gráficas que pueden aparecer en la pantalla de la computadora están dictadas por el tipo de tarjeta de video que ésta tenga.

2.- Algunas tarjetas de video pueden emular a otras y operar en varios modos de video gráficos con diversas características.

3.- Las gráficas se manejan dividiendo la pantalla en pixels; la resolución indica el número total de pixels en sentido horizontal y vertical.

4.- Los pixels se identifican mediante coordenadas que comienzan partiendo de cero en la esquina superior izquierda y aumentan hacia la derecha y abajo.

5.- Cada pixel tiene un color que se indica con un número que empieza en cero, que corresponde al fondo, y que puede llegar hasta un máximo determinado.

6.- Todas las figuras, dibujos y gráficas que aparecen en la pantalla se forman mediante grupos de pixels que tienen el mismo color.

7.- Los colores que aparecen en la pantalla se pueden modificar cambiando sus matices de acuerdo con la paleta de colores disponible.

8.- Los lenguajes de programación proporcionan medios para trazar gráficas comunes, como son líneas y círculos, y para controlar otros aspectos gráficos.

2 EL PAQUETE GRAFICO BATCH-GRAP

2.1 INTRODUCCION AL PAQUETE BATCH-GRAP

Los comandos de este paquete se dividen en tres grupos: comandos de ESTADO, comandos de MOSTRADO y comandos VARIOS. Los comandos de estado fijan algún valor que será utilizado en los comandos posteriores, como sería el color de las líneas o el tipo de relleno; los comandos de mostrado muestran alguna gráfica o trazo en la pantalla, como serían líneas o figuras; y los comandos varios realizan alguna otra acción y generalmente son de uso avanzado, es decir, es conveniente contar con un poco de experiencia para utilizarlos.

Los comandos de estado son los siguientes: GRAPHICS, GRAPMODE, SETCOLOR, SETLINE, SETFILL, SETTEXT y SETPOS de los cuales GRAPMODE y SETCOLOR son de uso avanzado. Varios de estos comandos utilizan el parámetro COLOR el cual debe ir seguido por un número que indica un color determinado, o bien, por el nombre de un color de acuerdo con la siguiente tabla, la cual está basada en los colores estándar de las tarjetas de video EGA y VGA: BLACK (0), BLUE (1), GREEN (2), CYAN (3), RED (4), MAGENTA (5), BROWN (6), LGRAY (7), GRAY (8), LBLUE (9), LGREEN (10), LCYAN (11), LRED (12), LMAGENTA (13), YELLOW (14) y WHITE (15).

Los comandos de mostrado son: LINE, BOX, CURVE, POLYGON, FILL, OUTTEXT, CLEAR y PUTIMAGE, de los que sólo el último es de uso avanzado. Para indicar posiciones de la pantalla en estos comandos se utiliza el parámetro AT seguido por una coordenada formada por dos números enteros que marcan la columna y la línea, en ese orden, comenzando en 0 0 que indica la esquina superior izquierda de la pantalla y aumentando hacia la derecha y abajo hasta llegar al límite de la pantalla, que corresponde a la coordenada 639 479 en videos tipo VGA.

Todos los comandos anteriores, excepto LINE y FILL, muestran sus gráficas en una zona rectangular de la pantalla la cual se define con el parámetro AT que marca su esquina superior izquierda, en combinación con el parámetro TO que marca la esquina inferior derecha, o bien, con el parámetro SIZE que indica el tamaño horizontal y vertical de la zona. Si se dá el parámetro SIZE pero se omite el tamaño vertical la zona estará formada por un cuadrado perfecto.

Recordando que en MS-DOS se permite usar el signo de igual (=), la coma (,) y el punto y coma (;) como separadores, será conveniente incluír estos caracteres en el lugar que más nos agrade para separar los parámetros de los comandos BATCH-GRAP con el fin de aumentar la claridad de los mismos. Por ejemplo:

```
    SETLINE COLOR=RED; STYLE=1; WIDE
    LINE AT 200,50 TO 450,140
```

Si al ejecutar alguno de los comandos BATCH-GRAP se teclea incorrectamente un parámetro se mostrará en la pantalla uno de los siguientes mensajes: "Syntax error" ó "Number expected". El primer mensaje indica que un parámetro (como sería la palabra COLOR) está mal escrito o simplemente no se proporcionó, mientras que el segundo indica que, de acuerdo con la forma de uso del comando, debe seguir un número y éste se encuentra mal escrito o no se proporcionó.

Cuando se presenten estos errores se mostrará encerrada entre comillas la parte del comando en la que se detectó el error, lo que permitirá identificarlo de inmediato. Si desea revisar la forma correcta de uso de un comando éste debe ejecutarse dándole como parámetro el caracter ? (por ejemplo: LINE ?).

Si los parámetros están correctamente escritos el comando se ejecutará pero todavía se podrá presentar algún error de ejecución, en cuyo caso se mostrará un mensaje descriptivo en inglés. Para identificar la causa de este error se debe consultar la descripción de los errores de ejecución en este manual.

En la descripción de los comandos se utilizan las siguientes convenciones: se muestra en mayúsculas aquellas partes del comando que deben escribirse tal como aparecen, mientras que las palabras en minúsculas describen la información que el usuario debe proporcionar y que en la mayor parte de los casos debe ser un número. Cada parte del comando se muestra en la misma posición en la que debe ir, por lo que si hay varias palabras en un mismo lugar, colocadas unas arriba de otras, indicará que debe elegirse alguna de ellas y colocarla en ese lugar.

Los paréntesis cuadrados [y] encierran partes del comando que son opcionales, es decir, que se pueden inclufr u omitir dependiendo de lo que se desee; es importante recordar que los paréntesis nunca deben teclearse. Finalmente, cuando aparezcan tres puntos seguidos (...) indicará que la parte o partes del comando que le anteceden podrán repetirse varias veces en la misma forma.

2.2 LOS COMANDOS DEL PAQUETE BATCH-GRAP

```
GRAPHICS [ON [BGIPATH dir] [BUFFERSIZE bytes] [TEXTLINES líneas] [VGACOORDS]]
         [OFF]
```

Activa (ON) o desactiva (OFF) el sistema gráfico; para poder utilizar los demás comandos de este paquete el sistema gráfico se debe activar, y al terminar de usar las gráficas se debe desactivar. La opción TEXTLINES permite definir una zona de la pantalla en la que se podrán teclear los comandos BATCH-GRAP sin que éstos interfieran con las gráficas; la opción VGACOORDS permite usar las mismas coordenadas de la tarjeta de video VGA (640 x 480) aunque la tarjeta de video sea de otro tipo.

```
         ┌OFF ┐
GRAPMODE │ON  │
         └modo┘
```

Desactiva temporalmente el modo de gráficas (OFF), activa el modo gráfico anterior (ON) o activa el modo gráfico indicado. Cada tipo de tarjeta de video cuenta con varios modos de video gráficos los cuales pueden tener diferente resolución y número de colores simultáneos, de acuerdo con lo indicado en la tabla de tipos de tarjetas de video (verla adelante), por lo que en ciertas tarjetas de video (p.e. CGA) se debe cambiar el modo de video con este comando de acuerdo con las necesidades de graficación.

```
SETCOLOR [0:matiz0 1:matiz1 ... 15:matiz15]
```

Fija el aspecto visual de los colores utilizados. Cada pareja de números cambia la apariencia del color dado por un cierto matiz que se indica con un número entero que debe tener un valor entre 0 y 63, lo que permite 64 tonos entre el negro y el blanco. Al activar el sistema gráfico se tienen los siguientes matices estándar en los 16 colores de las tarjetas EGA/VGA: 0:0 1:1 2:2 3:3 4:4 5:5 6:20 7:7 8:56 9:57 10:58 11:59 12:60 13:61 14:62 15:63. El número de color 0 siempre indicará el fondo de la pantalla.

```
SETLINE [COLOR color] [STYLE línea] [WIDE] [XOR]
```

Fija el color de las líneas (COLOR), el tipo de línea (STYLE 0-4), si la línea será gruesa (WIDE) y si se trazará con la combinación XOR.

```
                   ┌TO col1 lin1       ┐
LINE [AT col0 lin0] │SIZE hor1 [ver1]   │ ... [FILL [NOBORDER]]
                   └POLAR rad1 [ang1]  ┘
```

Traza líneas rectas de uno a otro de los puntos dados. El parámetro POLAR permite trazar líneas mediante coordenadas polares indicando su radio y ángulo polares a partir del punto anterior; para fijar la posición o los valores del radio y ángulo polares sin trazar ninguna línea use el comando SETPOS. La opción FILL indica rellenar el interior de la figura trazada; la opción NOBORDER indica rellenar la figura sin trazar el perímetro.

```
BOX [AT izq arr] ┌TO der aba      ┐ [FILL [NOBORDER]]
                 └SIZE hor [ver]  ┘
```

Traza un rectángulo en la zona dada. La opción FILL indica rellenar su interior; la opción NOBORDER indica no trazar el perímetro.

```
CURVE [AT izq arr] ┌TO der aba    ┐ [ARC inicio fin] [FILL [NOBORDER]]
                   └SIZE hor [ver]┘
```

Traza una elipse delimitada por la zona dada. La opción ARC indica trazar sólo un arco de curva entre los grados dados los cuales empiezan en el eje horizontal y avanzan en sentido contrario al del reloj; la opción FILL indica rellenar el interior; la opción NOBORDER no trazará el perímetro.

```
POLYGON [AT i a] ┌TO de ab ┐ [SIDES lados] [ANGLE ang] [STAR] [FILL [NOBORDER]]
                 └SIZE h[v]┘
```

Traza un polígono inscrito a la elipse delimitada por la zona dada, con el número de lados dado. La opción ANGLE indica un ángulo de giro; la opción STAR indica trazar una estrella en vez de un polígono; la opción FILL pide rellenar el interior; la opción NOBORDER indica no trazar el perímetro.

```
SETFILL [COLOR color] ┌STYLE relleno  ┐
                      └SHADE densidad┘
```

Fija el color del relleno (COLOR) y el tipo de relleno, ya sea con un cierto patrón (STYLE 0-12) o con una densidad de sombreado (SHADE 0-10).

```
FILL [AT col lin] BORDER [color1]
                  CHANGETO color2
```

Rellena una zona de la pantalla comenzando en el punto dado. El parámetro BORDER indica rellenar la zona cerrada delimitada por una línea del color actual, o del color1 indicado, con el tipo de relleno actual; el parámetro CHANGETO indica cambiar el color de la zona formada por todos los pixels adyacentes que tengan el mismo color del punto inicial por el color2 dado.

```
SETTEXT [COLOR color] [STYLE letra] [CHARSIZE tamaño] [VERTICAL]
```

Fija el color de los caracteres (COLOR), su tipo de letra (STYLE 0-10), su tamaño inicial (CHARSIZE 1-10) y si se trazarán verticalmente (VERTICAL).

```
OUTTEXT [AT izq arr] ┌TO der aba       [CAPS]┐ "texto sin comillas["]
                     └SIZE hor [ver]         ┘ 'texto sin apóstrofe[']
```

Muestra en la zona dada el texto indicado. Si se dá la opción TO o SIZE el texto ocupará exactamente la zona dada excepto con el tipo de letra 0, sin dejar el espacio inferior de las letras minúsculas al dar la opción CAPS.

```
CLEAR [AT izq arr] ┌TO der aba    ┐
   [DEVICE]        └SIZE hor [ver]┘
```

Borra la zona indicada o toda la pantalla si se dá la opción DEVICE.

```
GETIMAGE [AT izq arr] ┌TO der aba    ┐ INTO archivo
                      └SIZE hor [ver]┘      BUFFER [num] [CLEAR]
```

Almacena la imagen gráfica de la zona dada en el archivo indicado (INTO) o en el buffer de memoria número 0 o indicado (BUFFER 0-9), borrando los demás buffers si se dá la opción CLEAR. Si se usa un buffer se tendrá un límite en el tamaño máximo de la imagen que será el 42% de una pantalla VGA de 640 x 480 pixels ó el 58% de una pantalla EGA de 640 x 350 pixels; en los demás videos no habrá límite. El archivo tendrá la extensión .PCX.

```
PUTIMAGE [AT izq arr] FROM archivo      [COMBINE combinación]
                      BUFFER [num]
```

Muestra en el punto dado la imagen gráfica antes almacenada con el comando GETIMAGE en el archivo indicado (FROM) o en el buffer de memoria número 0 o indicado (BUFFER 0-9). La opción COMBINE indica la forma en que se combinará la imagen por mostrar con el contenido actual de la pantalla y debe ir seguida por alguna de las siguientes opciones: XOR, OR, AND o NOT. El archivo debe tener la extensión .PCX y seguir este formato gráfico.

```
CHOICE AT izq1 arr1 TO der1 aba1 ┌AT izq2 arr2 TO der2 aba2┐ ... [INITIAL num]
                SIZE h1 [v1]     └             SIZE h2 [v2]┘
```

Permite hacer una selección gráfica. La primer zona, o la indicada en la opción INITIAL, se mostrará activada pudiéndose cambiar de zona con las teclas de flechas o con HOME y END; al oprimir ENTER se regresará el número de la zona actual por el ERRORLEVEL o un cero si se oprime ESC; al oprimir un dígito entre 1 y 9 se seleccionará directamente esa zona.

```
                              ┌TO col1 lin1     ┐
SETPOS ┌AT col0 lin0]       ┐ │SIZE hor1 [ver1] │
       └   POLAR rad [ang]  ┘ └POLAR rad [ang]  ┘
```

Fija la posición, tamaño y valores polares actuales que utilizará el siguiente comando que no incluya alguno de estos parámetros. La palabra POLAR colocada enseguida de AT permite cambiar la posición en forma polar.

2.3 INFORMACION ADICIONAL SOBRE ALGUNOS COMANDOS

Casi todos los comandos del paquete BATCH-GRAP que cambian el valor de algún parámetro numérico también guardan internamente el nuevo valor de tal forma que éste se convierte en un "valor actual" que podrá ser utilizado en la ejecución posterior de otro comando. De esta manera para tomar el valor actual de un cierto parámetro bastará con comenzar su número con un signo más o menos, el cual indicará tomar el valor almacenado y sumarle ó restarle el número dado.

Los parámetros que se almacenan en esta forma son los 16 matices fijados en SETCOLOR, los colores y tipos de línea, relleno y caracteres fijados en los comandos SETLINE, SETFILL y SETTEXT, el número de lados y el ángulo de giro dados en POLYGON, y el número del buffer de GETIMAGE y PUTIMAGE. Por ejemplo, el comando SETFILL COLOR +3 STYLE -1 fijaría el color del relleno tres colores adelante del color actual y el tipo de relleno al inmediato anterior al actual.

Las posiciones de la pantalla y los tamaños de las figuras fijadas con los parámetros AT y TO/SIZE también se almacenan de esta manera, sólo que en este caso no se almacena la posición exacta del parámetro AT sino la de la esquina superior derecha de la zona definida por ambos parámetros, pero si esta esquina rebasa el límite derecho de la pantalla se avanzará verticalmente y regresará al extremo izquierdo de la misma, y cuando rebase el fondo se regresará a la parte superior de la pantalla simulando el movimiento del cursor de texto. De esta manera si se muestran varias figuras sin los parámetros AT ni TO/SIZE cada una de ellas se mostrará a la derecha de la anterior y del mismo tamaño.

Las únicas excepciones a lo dicho anteriormente son los comandos GETIMAGE y CLEAR en los cuales la posición actual se fijará en su esquina superior izquierda, y el comando LINE el cual siempre fijará la posición actual en el extremo final de la última línea trazada. La posición y tamaño actuales se podrán fijar directamente sin trazar ninguna figura mediante el comando SETPOS. Los valores del radio y ángulo polares del parámetro POLAR también se podrán comenzar con un signo, con lo cual se tomarán a partir de los valores actuales fijados previamente en los comandos LINE o SETPOS con este mismo parámetro.

GRAPHICS [ON [BGIPATH dir] [BUFFERSIZE bytes] [TEXTLINES líneas] [VGACOORDS]]
GRAPHIC [OFF]

Cuando el sistema gráfico se active el video se cambiará a un modo gráfico y al desactivarlo se regresará al modo de texto original, pero la pantalla se podrá cambiar por un momento al modo de texto normal sin necesidad de desactivar todo el sistema con el comando GRAPMODE. La opción BGIPATH permite indicar el directorio en el que se encuentran los archivos .BGI y .CHR; si esta opción no se proporciona estos archivos deben estar en disco en el directorio actual, o bien, si se cuenta con la versión 3 o superior de MS-DOS, en el mismo directorio del programa GRAPHICS.EXE. La opción BUFFERSIZE permite cambiar el tamaño estándar del buffer de memoria del sistema gráfico; esta opción debe incluírse con un valor mayor a 4096 si se presenta el error de ejecución "Out of memory in scan/flood fill".

2.4 ERRORES DE EJECUCION DE LOS COMANDOS GRAFICOS

BGI Error: Not in graphics mode
El video no se encuentra en modo gráfico. Para poder mostrar gráficas en la pantalla el modo gráfico debe recuperarse con GRAPMODE ON.

Device driver file not found
El archivo .BGI indicado no está en disco. Para poder activar el sistema este archivo debe estar en el directorio indicado por la opción BGIPATH.

Graphics error
Valor inválido de un parámetro. Revise que los valores proporcionados con los parámetros no excedan los límites máximos permitidos.

Graphics hardware not detected
La tarjeta de video de la computadora no maneja gráficas o no es una tarjeta estándar, por lo que el sistema gráfico no puede activarse.

Graphics I/O error
Error en entrada/salida. Revise la unidad de discos e intente de nuevo.

Graphics not installed (use GRAPHICS ON)
El sistema gráfico no ha sido instalado. Para poder ejecutar cualquier comando BATCH-GRAP el sistema gráfico debe activarse con GRAPHICS ON.

Invalid device driver file
El archivo .BGI indicado es inválido. Por alguna razón este archivo ha sido alterado; vuelva a copiarlo del disco original de este paquete.

Invalid font file
El archivo .CHR indicado es inválido. Por alguna razón este archivo ha sido alterado; vuelva a copiarlo del disco original de este paquete.

Invalid font number
Tipo de letra inválido. Revise que el tipo de letra esté entre 0 y 10.

Invalid graphics mode for selected driver
El modo de video solicitado no es válido para esta tarjeta. Revise los modos de video válidos para esta tarjeta de video.

Invalid version number
Por alguna razón los archivos *.BGI o *.CHR del sistema no corresponden con los adecuados; vuelva a copiarlos del disco original de este paquete.

Not enough memory to get image
No hay suficiente memoria disponible para tomar la imagen gráfica. Aumente la memoria disponible e intente de nuevo.

Not enough memory to load driver
No hay suficiente memoria disponible para activar el sistema. Aumente la memoria disponible e intente de nuevo.

Not enough memory to load font

No hay suficiente memoria disponible para activar el tipo de letra. Aumente la memoria disponible e intente de nuevo.

Number expected

Un número está mal escrito, o debe ir y no se proporcionó.

Out of memory in flood fill
Out of memory in scan fill

No hay suficiente memoria para rellenar. Desactive y vuelva a activar el sistema dando la opción BUFFERSIZE con un número cada vez mayor, volviendo a intentar la operación de relleno hasta que ya no se presente el error.

Syntax error

Un parámetro está mal escrito, o debe ir y no se proporcionó. Revise el orden en que deben proporcionarse los parámetros; recuerde que un parámetro que no está marcado como [opción] siempre debe proporcionarse.

2.5 TABLAS DE VALORES DIVERSOS DEL SISTEMA GRAFICO

TIPOS DE LINEA PARA LA OPCION STYLE DEL COMANDO SETLINE.

0	Línea continua (línea estándar).
1	Línea de puntos (líneas cortas, espacios largos).
2	Líneas y espacios de igual longitud.
3	Línea largas, espacios cortos.
4	Línea oculta (línea, punto, ...).

TIPOS DE RELLENO PARA LA OPCION STYLE DEL COMANDO SETFILL.

0	Rellena con el color del fondo (implica borrar).
1	Relleno sólido continuo (relleno estándar).
2	Líneas horizontales.
3	Diagonales delgadas hacia la derecha.
4	Diagonales gruesas hacia la derecha.
5	Diagonales gruesas hacia la izquierda.
6	Diagonales delgadas hacia la izquierda.
7	Cuadrícula horizontal chica.
8	Cuadrícula diagonal grande.
9	Sombreado grueso.
10	Sombreado fino.
11	Sombreado mediano.
12	Cuadros pequeños.

PORCENTAJES DE DENSIDADES DE SOMBREADO PARA LA OPCION SHADE
Y PARA LA COMBINACION DE DOS COLORES, AMBAS FIJADAS CON EL COMANDO SETFILL.

0: 1.57%	1: 6.25%	2: 12.5%	3: 25%	4: 37.5%	5: 50%
10: 98.43%	9: 93.75%	8: 87.5%	7: 75%	6: 62.5%	

TIPOS DE LETRA PARA LA OPCION STYLE DEL COMANDO SETTEXT.

Núm. de letra	Archivo .CHR	Tipo de letra
0	(Ninguno)	Letra normal (tipo estándar).
1	TRIP.CHR	Letra Triplex.
2	LITT.CHR	Letra pequeña.
3	SANS.CHR	Letra Sansserif.
4	GOTH.CHR	Letra gótica.
5	SCRI.CHR	Letra manuscrita.
6	SIMP.CHR	Letra Simplex.
7	TSCR.CHR	Letra itálica.
8	LCOM.CHR	Letra romana.
9	EURO.CHR	Letra grande.
10	BOLD.CHR	Letra Bold.

TIPOS DE TARJETAS DE VIDEO Y MAXIMO MODO DE VIDEO, RESOLUCION Y COLORES.

En la siguiente tabla se muestra, para cada tarjeta de video, el nombre del archivo .BGI necesario para activar el sistema, el número de su modo de video estándar, y el máximo número de columna, línea y color de ese modo de video.

Núm. de tarjeta	Tarjeta de video	Archivo .BGI	Máximo modo	Máxima columna	Máxima línea	Máximo color
1	CGA	CGA.BGI	4	639	199	1
2	MCGA	CGA.BGI	5	639	479	1
3	EGA	EGAVGA.BGI	1	639	349	15
4	EGA-64	EGAVGA.BGI	1	639	349	3
5	EGA-MONO	EGAVGA.BGI	0	639	349	1
6	IBM-8514	IBM8514.BGI	1	1023	769	255
7	HERCULES	HERC.BGI	0	719	347	1
8	ATT-400	ATT.BGI	5	639	399	1
9	VGA	EGAVGA.BGI	2	639	479	15
10	PC-3270	PC3270.BGI	0	719	349	1

Los colores de las líneas, relleno y caracteres se indican mediante números enteros que empiezan en 0 y que pueden llegar hasta el valor máximo mostrado en la tabla anterior; el número de color 0 siempre indica el fondo de la pantalla.

En forma similar las posiciones en la pantalla empiezan en 0 y pueden llegar hasta los valores máximos mostrados en la tabla anterior, pero si se utiliza la opción VGACOORDS del comando GRAPHICS los límites de las coordenadas serán siempre los mismos, correspondiendo a los de la tarjeta de video VGA.

Igualmente, se cuenta con varios modos de video los cuales empiezan en 0 y llegan hasta el modo máximo mostrado en la tabla anterior. El modo de video se puede cambiar con el comando GRAPMODE lo cual cambiará su máximo número de columna y línea y, dependiendeo del tipo de tarjeta, también su máximo color.

Por ejemplo, los modos del 0 al 3 de las tarjetas CGA, MCGA y ATT-400 tienen una resolución menor de 320 x 200 pixels, pero muestran 4 colores simultáneos formados por el color del fondo más las siguientes combinaciones; modo 0: verde claro, rojo claro y amarillo; modo 1: cyan claro, morado claro y blanco; modo 2: verde, rojo y café; modo 3: cyan, morado y gris. En estas tarjetas se podrán cambiar los colores de la pantalla con un comando SETCOLOR que incluya valores sólo para los números de color 0 y 1; el número 0 indicará el color del fondo de la pantalla y se le podrá asignar un valor estándar entre 0 y 15, el número 1 indicará los demás colores y se le podrá asignar un valor entre 0 y 3 el cual corresponderá a las combinaciones de tres colores antes descritas.

NOTA: Las tarjetas de video tipo Hércules no pueden mostrar en la pantalla los caracteres normales de texto cuando están en el modo gráfico, lo que impide observar los comandos que se ejecuten en forma interactiva a pesar de que éstos se ejecutarán correctamente si están bien escritos. La mejor forma de usar gráficas con tarjetas tipo Hércules es mediante archivos BATCH que activen el modo gráfico, muestren algunas gráficas y regresen a texto al terminar.

```
{ DEFINICIONES PARA TURBO-PASCAL }

const
  MaxColors = 15;

type
  ArcCoordsType = record
                    X, Y: integer;
                    XStart, YStart: integer;
                    XEnd, YEnd: integer;
                  end;
type
  FillPatternType = array [1..8] of byte;

type
  FillSettingsType = record
                       Pattern: word;
                       Color: word;
                     end;

type
  LineSettingsType = record
                       LineStyle: word;
                       Pattern: word;
                       Thickness: word;
                     end;

type
  PaletteType = record
                  Size: byte;
                  Colors: array [0..MaxColors] of shortint;
                end;

type
  TextSettingsType = record
                       Font: word;
                       Direction: word;
                       CharSize: word;
                       Horiz: word;
                       Vert: word;
                     end;

type
  ViewPortType = record
                   Left, Top, Right, Bottom: integer;
                   Clip: boolean;
                 end;
```

```
/* DEFINICIONES PARA TURBO-C */

#define MAXCOLORS 15

struct arccoordstype {
 int x, y;
 int xstart, ystart, xend, yend;
};

struct fillsettingstype {
 int pattern;
 int color;
};

struct linesettingstype {
 int linestyle;
 unsigned upattern;
 int thickness;
};


struct palettetype {
 unsigned char size;
 signed char colors[MAXCOLORS+1];
};

struct textsettingstype {
 int font;
 int direction;
 int charsize;
 int horiz;
 int vert;
};

struct viewporttype {
 int left, top, right, bottom;
 int clip;
};
```

DESCRIPCION DE LAS FUNCIONES DEL SISTEMA GRAFICO BGI (Pascal, c, CLIPPER)

```
Arc(X, Y: integer; StAngle, EndAngle, Radius: word);
void arc(int x, int y, int stangle, int endangle, int radius);
ARC(X, Y, STANGLE, ENDANGLE, RADIUS)
```

Traza un arco de círculo con centro en (X,Y), de STANGLE a ENDANGLE grados y radio RADIUS (Ver CIRCLE).

```
Bar(Left, Top, Right, Bottom: integer);
void bar(int left, int top, int right, int bottom);
BAR(LEFT, TOP, RIGHT, BOTTOM)
```

Dibuja un rectángulo relleno sin trazar su perímetro (Ver RECTANGLE).

```
Bar3D(Left, Top, Right, Bottom: integer; Depth: word; Top: boolean);
void bar3d(int left, int top, int right, int bottom, int depth, int topflag);
BAR3D(LEFT, TOP, RIGHT, BOTTOM, DEPTH, TOPFLAG)
```

Traza y rellena un rectángulo con una proyección de 3ra. dimensión de profundidad DEPTH pixels en un lado, y también arriba si TOPFLAG es <> 0.

```
ERRORCODE = CHANGEFILL(X, Y, COLOR)
```

Cambia el color de la zona que empieza en las coordenadas dadas y que incluye todo pixel adyacente que tenga el mismo color del punto inicial; regresa un código de estatus similar al de la función GRAPHRESULT (Ver FLOODFILL). TURBO: Esta función no forma parte del sistema BGI.

```
Circle(X, Y: integer; Radius: word);
void circle(int x, int y, int radius);
CIRCLE(X, Y, RADIUS)
```

Traza un círculo con centro en (X,Y) y radio RADIUS (Ver ELLIPSE).

```
ClearDevice;
void cleardevice(void);
CLEARDEVICE()
```

Borra toda la pantalla y mueve el cursor gráfico a (0,0).

```
ClearViewPort;
void clearviewport(void);
CLEARVIEWPORT()
```

Borra el contenido del viewport activo y mueve el cursor gráfico a (0,0).

```
CloseGraph;
void closegraph(void);
CLOSEGRAPH()
```

Desactiva el sistema gráfico (Ver INITGRAPH).

```
DetectGraph(var GraphDriver, GraphMode: integer);
void detectgraph(int *graphdriver,int *graphmode);
GRAPHDRIVER = 0  && CUALQUIER VALOR ENTERO DE INICIALIZACION
GRAPHMODE = 0
DETECTGRAPH(@GRAPHDRIVER, @GRAPHMODE)
```

Determina el tipo de tarjeta instalada y su modo de video estándar.

```
DrawPoly(NumPoints: word; var PolyPoints);
void drawpoly(int numpoints, int *polypoints);
DECLARE POLYPOINTS[NUMPOINTS*2]  && [1]=X1, [2]=Y1, [3]=X2, [4]=Y2, ...
DRAWPOLY(NUMPOINTS, POLYPOINTS)
```
Traza una serie de líneas de uno a otro de los vértices (Ver FILLPOLY).

```
DRAWCURVE(X, Y, STANGLE, ENDANGLE, XRADIUS [,YRADIUS]  [,.FILL.])
```
Traza diversas líneas curvas conjuntando la operación de las funciones ARC, CIRCLE, ELLIPSE, FILLELLIPSE, PIESLICE y SECTOR de acuerdo con los parámetros proporcionados. TURBO: Esta función no forma parte del sistema BGI, sino que se diseñó para evitar los errores que presentan las funciones antes mencionadas (Ver la Descripción de Errores Detectados).

```
DRAWPOLYGON(X, Y, SIDES, XRADIUS [,YRADIUS] [,.STAR. [,.FILL.] [,ROTATE]])
```
Traza un polígono regular o irregular con centro en (X,Y), de SIDES lados, radio horizontal XRADIUS y vertical YRADIUS, trazará una estrella si .STAR. es .T., rellena el interior si .FILL. es .T., y se girará el número de grados de ROTATE. TURBO: Esta función no es del sistema BGI.

```
Ellipse(X, Y: integer; StAngle, EndAngle, XRadius, YRadius: word);
void ellipse(int x,int y, int stangle,int endangle, int xradius,int yradius);
ELLIPSE(X, Y, STANGLE, ENDANGLE, XRADIUS, YRADIUS)
```
Traza un arco de elipse con centro en (X,Y), de STANGLE a ENDANGLE grados, radio horizontal XRADIUS y vertical YRADIUS (Ver FILLELLIPSE)

```
FillEllipse(X, Y: integer; XRadius, YRadius: word);
void fillellipse(int x, int y, int xradius, int yradius);
FILLELLIPSE(X, Y, XRADIUS, YRADIUS)
```
Traza y rellena una elipse con centro en (X,Y), radio horizontal XRADIUS y vertical YRADIUS (Ver PIESLICE).

```
FillPoly(NumPoints: word; var PolyPoints);
void fillpoly(int numpoints, int *polypoints);
DECLARE POLYPOINTS[NUMPOINTS*2]  && [1]=X1, [2]=Y1, [3]=X2, [4]=Y2, ...
FILLPOLY(NUMPOINTS, POLYPOINTS)
```
Traza una serie de líneas de uno a otro de los vértices indicados y rellena el interior de la figura así formada.

```
FloodFill(X, Y: integer; Border: word);
void floodfill(int x, int y, int border);
FLOODFILL(X, Y, BORDER)
```
Rellena una figura delimitada por una línea continua del color BORDER. Esta función no opera con la tarjeta de video IBM-8514.

```
GetArcCoords(var ArcCoords: ArcCoordsType);
void getarccoords(struct arccoordstype *arccoords);
DECLARE ARCCOORDS[6]  && 1=X, 2=Y, 3=XSTART, 4=YSTART, 5=XEND, 6=YEND
GETARCCOORDS(ARCCOORDS)
```
Obtiene las coordenadas del centro y extremos del arco trazado por la última función ARC o ELLIPSE invocada.

```
GetAspectRatio(var XAsp, YAsp: word);
void getaspectratio(int *xasp, int *yasp);
XASP = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
YASP = 0
GETASPECTRATIO(@XASP, @YASP)
```
Obtiene el tamaño visual relativo de los pixels del modo de video actual. Regresa los valores YASP=10000 y XASP<10000 excepto en las tarjetas VGA e IBM-8514 las cuales muestran pixels del mismo tamaño horizontal:vertical.

```
GetBkColor : word;
int getbkcolor(void);
BKCOLOR = GETBKCOLOR()
```
Obtiene el color actual del fondo.

```
GetColor : word;
int getcolor(void);
COLOR = GETCOLOR()
```
Obtiene el color actual de líneas.

```
GetDefaultPalette(var Palette: PaletteType);
struct palettetype * getdefaultpalette(void);
```
Obtiene los colores gráficos estándar de las tarjetas EGA y VGA (Ver SETPALETTE). CLIPPER: Esta función no puede utilizarse desde Clipper.

```
GetDriverName : string;
char * getdrivername(void);
DRIVERNAME = GETDRIVERNAME()
```
Obtiene el nombre de la tarjeta de gráficas activa.

```
GetFillPattern(var FillPattern: FillPatternType);
void getfillpattern(char *fillpattern);
FILLPATTERN = SPACE(8)  && SECUENCIA DE 8 CARACTERES DE INICIALIZACION
GETFILLPATTERN(@FILLPATTERN)
```
Obtiene el patrón de relleno definido por el usuario (Ver SETFILLPATTERN)

```
GetFillSettings(var FillSettings: FillSettingsType);
void getfillsettings(struct fillsettingstype *fillsettings);
DECLARE FILLSETTINGS[2]  && 1=FILLSTYLE, 2=COLOR
GETFILLSETTINGS(FILLSETTINGS)
```
Obtiene el tipo y color actuales del relleno (Ver SETFILLSTYLE). Si el relleno se estableció con SETFILLPATTERN, FILLSTYLE valdrá 12.

```
GetGraphMode : integer;
int getgraphmode(void);
GRAPHMODE = GETGRAPHMODE()
```
Obtiene el modo de video actual (Ver INITGRAPH).

```
GetImage(Left, Top, Right, Bottom: integer; var BitMap);
void getimage(int left, int top, int right, int bottom, void *bitmap);
BITMAP = SPACE(IMAGESIZE(LEFT,TOP,RIGHT,BOTTOM))
GETIMAGE(LEFT, TOP, RIGHT, BOTTOM, @BITMAP)
```
Toma una imagen gráfica y la guarda en BITMAP (Ver IMAGESIZE y PUTIMAGE).

```
GetLineSettings(var LineSettings: LineSettingsType);
void getlinesettings(struct linesettingstype *linesettings);
DECLARE LINESETTINGS[3]  && 1=LINESTYLE, 2=PATTERN, 3=THICKNESS
GETLINESETTINGS(LINESETTINGS)
     Obtiene el tipo, patrón y ancho de la línea actual (Ver SETLINESTYLE).

GetMaxColor : word;
int getmaxcolor(void);
MAXCOLOR = GETMAXCOLOR()
     Obtiene el máximo número de color disponible en el modo de video actual.

GetMaxMode : word;
int getmaxmode(void);
MAXMODE = GETMAXMODE()
     Obtiene el máximo modo de video gráfico de la tarjeta de video activa.

GetMaxX : integer;
int getmaxx(void);
MAXX = GETMAXX()
     Obtiene la máxima coordenada horizontal del modo de video actual.

GetMaxY : integer;
int getmaxy(void);
MAXY = GETMAXY()
     Obtiene la máxima coordenada vertical del modo de video actual.

GetModeName(ModeNumber: integer) : string;
char * getmodename(int modenumber);
MODENAME = GETMODENAME(MODENUMBER)
     Obtiene el nombre del modo de video indicado de la tarjeta activa.

GetModeRange(GraphDriver: integer; var LoMode, HiMode: integer);
void getmoderange(int graphdriver, int *lomode, int *himode);
LOMODE = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
HIMODE = 0
GETMODERANGE(GRAPHDRIVER, @LOMODE, @HIMODE)
     Obtiene el rango de los modos de video gráficos disponibles  en la tarjeta
     de video indicada. Si GRAPHDRIVER es -1 se tomará la tarjeta activa.

GetPalette(var Palette: PaletteType);
void getpalette(struct palettetype *palette);
PALETTE = SPACE(17)  && 1=SIZE, 2=COLOR0, 3=COLOR1, ..., 17=COLOR15
GETPALETTE(@PALETTE)
     Obtiene los colores gráficos actuales de las tarjetas EGA y VGA.

GetPaletteSize : integer;
int getpalettesize(void);
PALETTESIZE = GETPALETESIZE()
     Obtiene el número de colores que se  pueden modificar.  CLIPPER: El nombre
     de esta función se cambió para evitar conflictos con la función anterior.
```

```
GetPixel(X, Y: integer) : word;
unsigned getpixel(int x, int y);
PIXEL = GETPIXEL(X, Y)
```
Obtiene el color actual del pixel indicado.

```
GetTextSettings(var TextSettings: TextSettingsType);
void gettextsettings(struct textsettingstype *textsettings);
DECLARE TEXTSETTINGS[5]  && 1=FONT, 2=DIRECTION, 3=CHARSIZE, 4=HORIZ, 5=VERT
GETTEXTSETTINGS(TEXTSETTINGS)
```
Obtiene los parámetros del texto (Ver SETTEXTSTYLE y SETTEXTJUSTIFY).

```
GetViewSettings(var ViewSettings: ViewPortType);
void getviewsettings(struct viewporttype *viewsettings);
DECLARE VIEWSETTINGS[5]  && 1=LEFT, 2=TOP, 3=RIGHT, 4=BOTTOM, 5=CLIP
GETVIEWSETTINGS(VIEWSETTINGS)
```
Obtiene los parámetros del viewport activo (Ver SETVIEWPORT).

```
GetX : integer;
int getx(void);
X = GETX()
```
Obtiene la coordenada horizontal actual del cursor gráfico.

```
GetY : integer;
int gety(void);
Y = GETY()
```
Obtiene la coordenada vertical actual del cursor gráfico.

```
GraphDefaults;
void graphdefaults(void);
GRAPHDEFAULTS()
```
Fija los parámetros gráficos a sus valores estándar: cancela el viewport, mueve el cursor gráfico a (0,0), fija los colores estándar, los tipos de línea y relleno estándar, y el texto y justificaciones estándar.

```
GraphErrorMsg(ErrorCode: integer) : string;
char * grapherrormsg(int errorcode);
ERRORMSG = GRAPHERRORMSG(ERRORCODE)
```
Obtiene el mensaje de error del código indicado (Ver GRAPHRESULT).

```
GraphResult : integer;
int graphresult(void);
ERRORCODE = GRAPHRESULT()
```
Regresa el estatus de la última operación gráfica; un cero indica "operación correcta" mientras que un valor negativo indica algún error:

-1=Sistema BGI no inicializado	-2=No existe una tarjeta gráfica
-3=Archivo .BGI no encontrado	-4=Archivo .BGI inválido
-5=Falta memoria para el .BGI	-6=Falta memoria para trazar
-7=Falta memoria para rellenar	-8=Archivo .CHR no encontrado
-9=Falta memoria para el .CHR	-10=Modo de video inválido
-11=Error interno (valor inválido)	-12=Error en entrada/salida
-13=Archivo .CHR inválido	-14=Tipo de letra inválido
-15=Dispositivo inválido	-18=Archivo de una versión inválida

```
ImageSize(Left, Top, Right, Bottom: integer) : word;
unsigned imagesize(int left, int top, int right, int bottom);
SIZE = IMAGESIZE(LEFT, TOP, RIGHT, BOTTOM)
     Obtiene el tamaño en bytes del área que requiere  la  función GETIMAGE; si
     el tamaño es mayor o igual a 64K,  se regresará un 0  (y no un -1 [0xFFFF]
     como indica el manual; ver la Descripción de Errores Detectados).

InitGraph(var GraphDriver, GraphMode: integer; PathToDriver: string);
void initgraph(int *graphdriver, int *graphmode, char *pathtodriver);
GRAPHDRIVER = 0 && NUMERO DE TARJETA, 0 = AUTODETECCION
GRAPHMODE = 0  && MODO INICIAL DE VIDEO SI GRAPHDRIVER <> 0
INITGRAPH(@GRAPHDRIVER, @GRAPHMODE, "PATHTODRIVER")
     Inicializa el sistema gráfico.  Si GRAPHDRIVER =  0 invocará a DETECTGRAPH
     para identificar la tarjeta instalada y su modo  de video  estándar, de lo
     contrario activará la tarjeta y el  modo de video indicados.  Los archivos
     *.BGI y *.CHR se buscarán en el directorio indicado por PATHTODRIVER, para
     activar el  sistema se  requiere  el  archivo  *.BGI  correspondiente. Las
     tarjetas y modos de video soportados por este sistema son las siguientes:
     1=CGA:       modos 0, 1, 2 y 3 = 320 x 200 a 4 colores (ver SETCGAPALETTE)
                  modo 4 = 640 x 200 a 1 color (SETBKCOLOR cambia) sobre negro.
     2=MCGA:      mismos modos CGA + modo 5 = 640 x 480 a 1 color sobre negro.
     3=EGA:       0 = 640 x 200 a 16 colores; 1 = 640 x 350 a 16 colores.
     4=EGA64:     0 = 640 x 200 a 16 colores; 1 = 640 x 350 a 4 colores.
     5=EGAMONO:   3 = 640 x 350 a 1 color (el del monitor) sobre negro.
     6=IBM8514:   0 = 640 x 480 a 256 colores; 1 = 1024 x 768 a 256 colores.
     7=HERCULES:  0 = 720 x 348 a 1 color (el del monitor) sobre negro.
     8=ATT400:    mismos modos CGA + modo 5 = 640 x 400 a 1 color sobre negro.
     9=VGA:       mismos modos EGA + modo 2 = 640 x 480 a 16 colores.
     10=PC3270:   0 = 720 x 350 a 1 color (el del monitor) sobre negro.
     En todos los casos el modo de video estándar es el de máxima resolución.

InstallUserDriver(Name: string; AutoDetectPtr: pointer) : integer;
int installuserdriver(char *name, int huge (*detect)(void));
     Instala un driver de un dispositivo de  video adicional para  ser manejado
     por el sistema. CLIPPER: Esta función no puede utilizarse desde Clipper.

InstallUserFont(FontFileName: string) : integer;
int installuserfont(char *name);
FONT = INSTALLFONT("FONTFILE.CHR")
     Instala un tipo de letra adicional a los  cuatro  proporcionados. CLIPPER:
     El nombre de esta función se cambió para evitar conflictos.

Line(X1, Y1, X2, Y2: integer);
void line(int x1, int y1, int x2, int y2);
LINE(X1, Y1, X2, Y2)
     Traza una línea de (X1,Y1) a (X2,Y2).

LineRel(Dx, Dy: integer);
void linerel(int dx, int dy);
LINEREL(DX, DY)
     Traza una línea del cursor gráfico a un punto situado a  (DX,DY) pixels de
     distancia, y mueve el cursor gráfico a la nueva posición.
```

```
LineTo(X, Y: integer);
void lineto(int x, int y);
LINETO(X, Y)
     Traza una línea del cursor gráfico al punto (X,Y), y mueve el cursor.

MoveRel(Dx, Dy: integer);
void moverel(int dx, int dy);
MOVEREL(DX, DY)
     Mueve el cursor gráfico a un punto situado a (DX,DY) pixels de distancia.

MoveTo(X, Y: integer);
void moveto(int x, int y);
MOVETO(X, Y)
     Mueve el cursor gráfico al punto (X,Y).

OutText(TextString: string);
void outtext(char *textstring);
OUTTEXT("TEXTSTRING")
     Muestra el texto indicado en la posición actual del cursor gráfico; si la
     dirección fijada en SETTEXTSTYLE es horizontal y la justificación
     horizontal fijada en SETTEXTJUSTIFY es a la izquierda, moverá la posición
     del cursor gráfico al final del texto (Ver OUTTEXTXY).

OutTextXY(X, Y: integer; TextString: string);
void outtextxy(int x, int y, char *textstring);
OUTTEXTXY(X, Y, "TEXTSTRING")
     Muestra el texto en la posición dada (Ver SETTEXTSTYLE y SETTEXTJUSTIFY).

PieSlice(X, Y: integer; StAngle, EndAngle, Radius: word);
void pieslice(int x, int y, int stangle, int endangle, int radius);
PIESLICE(X, Y, STANGLE, ENDANGLE, RADIUS)
     Traza y rellena una "rebanada de pastel" circular con centro en (X,Y), de
     STANGLE a ENDANGLE grados, y radio RADIUS (Ver SECTOR).

PutImage(Left, Top: integer; var BitMap; BitOp: word);
void putimage(int left, int top, void *bitmap, int bitop);
PUTIMAGE(LEFT, TOP, "BITMAP", BITOP)
     Muestra en la pantalla una imagen rectangular tomada con GETIMAGE. BITOP
     indica una combinación de la imagen con el contenido actual de la pantalla
     de acuerdo con: 0=COPY, 1=XOR, 2=OR, 3=AND, 4=COPY NOT.

PutPixel(X, Y: integer; Color: word);
void putpixel(int x, int y, int color);
PUTPIXEL(X, Y, COLOR)
     Cambia el color del pixel indicado.

Rectangle(Left, Top, Right, Bottom: integer);
void rectangle(int left, int top, int right, int bottom);
RECTANGLE(LEFT, TOP, RIGHT, BOTTOM)
     Traza un rectángulo (Ver BAR y BAR3D).
```

```
RegisterBGIDriver(Driver: pointer) : integer;
int registerbgidriver(void (*driver)(void));
```
Registra un manejador de tarjeta de video, convertido con BGIOBJ y ligado en el código, para ser utilizado en el sistema gráfico. Esta función debe invocarse antes de activar el sistema con INITGRAPH. CLIPPER: Esta función no puede utilizarse desde Clipper.

```
RegisterBGIFont(Font: pointer) : integer;
int registerbgifont(void (*font)(void));
```
Registra una definición de un tipo de letra, convertida con BGIOBJ y ligada en el código, para ser utilizada en el sistema gráfico. Esta función debe invocarse antes de activar el sistema con INITGRAPH. CLIPPER: Esta función no puede utilizarse desde Clipper.

```
RegisterFarBGIDriver(Driver: pointer) : integer;
int registerfarbgidriver(void far *driver);
DRIVER_STRING = MEMOREAD("DRIVER_FILE.BGI")
DRIVER = REGFARBGIDRIVER(@DRIVER_STRING)
```
Registra un manejador de tarjeta de video, convertido con BGIOBJ /F y ligado en el código, o leído previamente de un archivo *.BGI, para ser utilizado en el sistema gráfico. Esta función debe invocarse antes de activar el sistema con INITGRAPH. CLIPPER: El nombre de esta función se cambió para evitar conflictos con la función REGISTERFARBGIFONT.

```
RegisterFarBGIFont(Font: pointer) : integer;
int registerfarbgifont(void far *font);
FONT_STRING = MEMOREAD("FONT_FILE.CHR")
FONT = REGFARBGIFONT(@FONT_STRING)
```
Registra una definición de un tipo de letra, convertida con BGIOBJ /F y ligada en el código, o leída previamente de un archivo *.CHR, para ser utilizada en el sistema gráfico. Esta función debe invocarse antes de activar el sistema con INITGRAPH. CLIPPER: El nombre de esta función se cambió para evitar conflictos con la función REGISTERFARBGIDRIVER.

```
FONT_STRING = MEMOREAD("FONT_FILE.CHR")
FONT = REGISTERFONT(@FONT_STRING)
```
Copia la string dada a una zona permanente de memoria y la registra con la función REGFARBGIFONT. TURBO: Esta función no es del sistema BGI, se definió para evitar los problemas de manejo de memoria de Clipper 5.

```
RestoreCrtMode;
void restorecrtmode(void);
RESTORECRTMODE()
```
Regresa el video al modo de texto anterior a INITGRAPH sin desactivar el sistema gráfico. Para recuperar el modo gráfico utilice SETGRAPHMODE.

```
Sector(X, Y: integer; StAngle, EndAngle, XRadius, YRadius: word);
void sector(int x, int y, int stangle, int endangle, int xradius,int yradius);
SECTOR(X, Y, STANGLE, ENDANGLE, XRADIUS, YRADIUS)
```
Traza y rellena una "rebanada de pastel" elíptica con centro en (X,Y), de STANGLE a ENDANGLE grados, y radios XRADIUS y YRADIUS. (Ver PIESLICE).

```
SetActivePage(Page: word);
void setactivepage(int page);
SETACTIVEPAGE(PAGE)
```
Activa la página de video indicada para graficar, es decir, la página que recibirá las gráficas. Sólo las tarjetas de video Hércules, EGA y VGA (en sus modos de video 0 y 1) tienen dos páginas de video (Ver SETVISUALPAGE)

```
SetAllPalette(var Palette: PaletteType);
void setallpalette(struct palettetype *palette);
SETALLPALETTE("PALETTE")
```
Cambia los colores gráficos en las tarjetas EGA y VGA (Ver SETPALETTE).

```
SetAspectRatio(XAsp, YAsp: word);
void setaspectratio(int xasp, int yasp);
SETASPECTRATIO(XASP, YASP)
```
Cambia el tamaño visual horizontal:vertical relativo de los pixels utilizado por el sistema gráfico permitiendo corregir el trazado de círculos en un monitor de video desajustado (Ver GETASPECTRATIO).

```
SetBkColor(BkColor: word);
void setbkcolor(int color);
SETBKCOLOR(BKCOLOR)
```
Cambia el color del fondo de la pantalla. En los modos de video 4 y 5 de las tarjetas de video CGA, MCGA y ATT400 cambia el color de las líneas.

```
SETCGAPALETTE(PALETTE)
```
Cambia los colores gráficos en las tarjetas CGA, MCGA y ATT400. PALETTE debe ser un número entero entre 0 y 3 que define los colores asignados a los números de color 1, 2 y 3 en la misma forma que un modo de video, es decir: Modo 0: 1=verde claro, 2=rojo claro, 3=amarillo; Modo 1: 1=cyan claro, 2=magenta claro, 3=blanco; Modo 2: 1=verde, 2=rojo, 3=café; Modo 3: 1=cyan, 2=magenta, 3=gris. TURBO: Esta función no es del sistema BGI.

```
SetColor(Color: word);
void setcolor(int color);
SET_COLOR(COLOR)
```
Fija el color de las líneas. Los colores estándar de las tarjetas EGA y VGA son: 1=azul, 2=verde, 3=cyan, 4=rojo, 5=morado, 6=café, 7=gris claro, 8=gris oscuro, 9=azul claro, 10=verde claro, 11=cyan claro, 12=rojo claro 13=morado claro, 14=amarillo, 15=blanco. CLIPPER: Se cambió el nombre de esta función para evitar conflictos con la función homónima de Clipper.

```
SetFillPattern(FillPattern: FillPatternType; Color: word);
void setfillpattern(char *fillpattern, int color);
SETFILLPATTERN("FILLPATTERN", COLOR)
```
Fija el tipo del patrón de relleno al relleno modificable por el usuario definido en FILLPATTERN, el cual debe ser una serie de 8 bytes que define un cuadrado de 8 x 8 bits que corresponden a los pixels del patrón.

```
SetFillStyle(FillStyle, Color: word);
void setfillstyle(int fillstyle, int color);
SETFILLSTYLE(FILLSTYLE, COLOR)
```
Fija el tipo del patrón de relleno al relleno predefinido indicado por FILLSTYLE de acuerdo con la siguiente tabla: 0=Color del fondo (borrará), 1=Relleno sólido, 2=Líneas horizontales, 3=Diagonales / delgadas, 4=Diagonales / gruesas, 5=Diagonales \ delgadas, 6=Diagonales \ gruesas, 7=Cuadrícula pequeña, 8=Cuadrícula grande, 9=Sombreado grueso, 10=Sombreado ligero, 11=Sombreado intermedio.

```
SetGraphBufSize(BufSize: word);
unsigned setgraphbufsize(unsigned bufsize);
OLDBUFSIZE = SETGRAPHBUFSIZE(BUFSIZE)
```
Cambia el tamaño estándar del buffer interno del sistema gráfico (4K) lo cual permite utilizar menos memoria, o bien, evitar los errores -6 y -7. Esta función debe invocarse antes de activar el sistema con INITGRAPH.

```
SetGraphMode(Mode: integer);
void setgraphmode(int mode);
SETGRAPHMODE(MODE)
```
Cambia el modo de video activo al nuevo modo indicado por MODE de acuerdo con la tabla de modos de video mostrada en la función INITGRAPH. Permite regresar a un modo gráfico después de pasar a texto con RESTORECRTMODE.

```
SetLineStyle(LineStyle, Pattern, Thickness: word);
void setlinestyle(int linestyle, unsigned pattern, int thickness);
SETLINESTYLE(LINESTYLE, PATTERN, THICKNESS)
```
Fija el tipo de línea, patrón modificable y ancho de las líneas de acuerdo con lo siguiente. LINESTYLE: 0=Continua, 1=De puntos, 2=Segmentos cortos, 3=Segmentos largos, 4=Patrón definido por PATTERN, el cual debe ser un número entero de 16 bits; THICKNESS: 1=línea delgada, 3=gruesa.

```
SetPalette(ColorNum, Color: word);
void setpalette(int colornum, int color);
SETPALETTE(COLORNUM, COLOR)
```
Cambia un color gráfico en las tarjetas EGA y VGA. COLORNUM debe ser un número entero entre 0 y 15 que indica el número de color por cambiar, y COLOR debe ser un número entero entre 0 y 63 que define el nuevo color.

```
SetRGBPalette(ColorNum, Red, Green, Blue: integer);
void setrgbpalette(int colornum, int red, int green, int blue);
SETRGBPALETTE(COLORNUM, RED, GREEN, BLUE)
```
Cambia un color gráfico en la tarjeta IBM-8514. COLORNUM debe ser un número entero entre 0 y 255 que indica el número de color por cambiar; RED, GREEN y BLUE deben ser tres números enteros de 8 bits, de los que se tomarán sólo los 6 bits del lado izquierdo (mas significativos).

```
SetTextJustify(Horiz, Vert: word);
void settextjustify(int horiz, int vert);
SETTEXTJUSTIFY(HORIZ, VERT)
```
Fija la justificación horizontal y vertical del texto de acuerdo con lo siguiente. HORIZ: 0=Izquierda, 1=Centro, 2=Derecha; VERT: 0=Abajo, 2=Centro, 3=Arriba. (Ver SETTEXTSTYLE).

SetTextStyle(Font, Direction, CharSize: word);
void settextstyle(int font, int direction, int charsize);
SETTEXTSTYLE(FONT, DIRECTION, CHARSIZE)

Fija el tipo de letra, dirección y tamaño normal de los caracteres en la siguiente forma. FONT: 0=Estándar, 1=Triplex, 2=Pequeño, 3=Sansserif, 4=Gótico (1-4 requieren un archivo *.CHR); DIRECTION: 0=Horizontal, 1=Vertical; CHARSIZE: Número entero entre 0 y 10 (Ver SETUSERCHARSIZE).

SetUserCharSize(MultX, DivX, MultY, DivY: word);
void setusercharsize(int multx, int divx, int multy, int divy);
SETUSERCHARSIZE(MULTX, DIVX, MULTY, DIVY)

Fija un tamaño especial de los caracteres de los tipos de letra del 1 al 4; el ancho normal de los caracteres se multiplicará por MULTX/DIVX (ver TEXTWIDTH) y la altura por MULTY/DIVY (ver TEXTHEIGHT). Este tamaño se activará al invocar SETTEXTSTYLE con el parámetro CHARSIZE igual a cero.

SetViewPort(Left, Top, Right, Bottom: integer; Clip: boolean);
void setviewport(int left, int top, int right, int bottom, int clipflag);
SETVIEWPORT(LEFT, TOP, RIGHT, BOTTOM, CLIPFLAG)

Activa un viewport, es decir, una ventana a partir de la cual se trazarán todas las gráficas siguientes. Si CLIPFLAG es diferente de 0 las gráficas que desborden el viewport no se mostrarán, si es 0 sí aparecerán.

SetVisualPage(Page: word);
void setvisualpage(int page);
SETVISUALPAGE(PAGE)

Muestra en la pantalla la página de video indicada (Ver SETACTIVEPAGE).

SetWriteMode(WriteMode: integer);
void setwritemode(int mode);
SETWRITEMODE(WRITEMODE)

Fija el modo de trazado de líneas de las funciones LINE, LINEREL, LINETO, RECTANGLE y DRAWPOLY. Si WRITEMODE es 0 se trazarán normalmente, si es 1 se hará un XOR entre la línea y el contenido actual de la pantalla.

TextHeight(TextString: string) : word;
int textheight(char *textstring);
HEIGHT = TEXTHEIGHT("TEXTSTRING")

Obtiene la altura de un texto en pixels la cual depende del tipo de letra y tamaño fijados en SETTEXTSTYLE, y opcionalmente del factor MULTY/DIVY fijado en SETUSERCHARSIZE. La altura del texto se mide en dirección perpendicular a la fijada en SETTEXTSTYLE. (Ver OUTTEXT.)

TextWidth(TextString: string) : word;
int textwidth(char *textstring);
WIDTH = TEXTWIDTH("TEXTSTRING")

Obtiene el ancho de un texto en pixels el cual depende del tipo de letra y tamaño fijados en SETTEXTSTYLE, del número de caracteres de la string, y opcionalmente del factor MULTX/DIVX fijado en SETUSERCHARSIZE. El ancho del texto se mide en la dirección fijada en SETTEXTSTYLE. (Ver OUTTEXT.)

4 RELACION ENTRE EL SISTEMA BGI Y EL PAQUETE BATCH-GRAP

El paquete gráfico BATCH-GRAP fué desarrollado utilizando las capacidades del sistema BGI, específicamente las de Turbo-C++ segunda edición. Con el fin de hacer esto más aparente, en seguida se muestra una tabla que relaciona los comandos del paquete BATCH-GRAP con las funciones del sistema BGI utilizadas por cada uno de ellos; esta tabla puede ayudar a convertir un programa BATCH gráfico que use comandos BATCH-GRAP a su correspondiente programa en Turbo-C.

```
GRAPHICS [ON [BGIPATH dir] [BUFFERSIZE bytes] [TEXTLINES líneas] [VGACOORDS]]
         [OFF]
     DetectGraph, RegisterFarBGIDriver, RegisterFarBGIFont, SetGraphBufSize,
     InitGraph, GetDriverName, CloseGraph.

         ┌OFF ┐
GRAPMODE │ON  │
         └modo┘
     RestoreCRTMode, GetGraphMode, SetGraphMode, GetModeName.

SETCOLOR [0:matiz0 1:matiz1 ... 15:matiz15]
     GetBkColor, SetBkColor, GetPalette, GetPaletteSize, SetPalette.

SETLINE [COLOR color] [STYLE línea] [WIDE] [XOR]
     GetColor, SetColor, GetLineSettings, SetLineStyle, SetWriteMode.

                     ┌TO col1 lin1      ┐
LINE [AT col0 lin0]  │SIZE hor1 [ver1]  │ ... [FILL [NOBORDER]]
                     └POLAR rad1 [ang1] ┘
     Line, LineTo, LineRel, DrawPoly, FillPoly.

BOX [AT izq arr] ┌TO der aba      ┐ [FILL [NOBORDER]]
                 └SIZE hor [ver]  ┘
     Rectangle, Bar3D, Bar.

CURVE [AT izq arr] ┌TO der aba      ┐ [ARC inicio fin] [FILL [NOBORDER]]
                   └SIZE hor [ver]  ┘
     Circle, Ellipse, FillEllipse, Arc, GetArcCoords, PieSlice, Sector.

POLYGON [AT i a] ┌TO de ab ┐ [SIDES lados] [ANGLE ang] [STAR] [FILL [NOBORDER]]
                 └SIZE h[v]┘
     DrawPoly, FillPoly.

SETFILL [COLOR color] ┌STYLE relleno   ┐
                      └SHADE densidad  ┘
     GetFillSettings, GetFillPattern, SetFillStyle, SetFillPattern.

FILL [AT col lin] BORDER [color1]
                  CHANGETO color2
     FloodFill, GetColor, GetPixel, PutPixel.

SETTEXT [COLOR color] [STYLE letra] [CHARSIZE tamaño] [VERTICAL]
     GetColor, SetColor, GetTextSettings, SetTextStyle, SetTextJustify.
```

```
OUTTEXT [AT izq arr] ┌TO der aba      [CAPS]┐ "texto sin comillas["]
                     └SIZE hor [ver]        ┘ 'texto sin apóstrofe[']
     OutText, OutTextXY, TextWidth, TextHeight, SetUserCharSize.

CLEAR [AT izq arr] ┌TO der aba     ┐
      [DEVICE] .   └SIZE hor [ver] ┘
     GetViewSettings, SetViewPort, ClearViewPort, ClearDevice.

GETIMAGE [AT izq arr] ┌TO der aba     ┐ INTO archivo
                      └SIZE hor [ver] ┘      BUFFER [num] [CLEAR]
     ImageSize, GetImage.

PUTIMAGE [AT izq arr] FROM archivo       [COMBINE combinación]
                           BUFFER [num]
     PutImage.

                                  ┌TO col1 lin1     ┐
SETPOS ┌AT col0 lin0]       ┐     │SIZE hor1 [ver1] │
       └     POLAR rad [ang]┘     └POLAR rad [ang]  ┘
     MoveTo, MoveRel.
```

Otras funciones BGI utilizadas en forma general:
GetAspectRatio, GetMaxColor, GetMaxX, GetMaxY, GetX, GetY, GraphResult, GraphErrorMsg.

Funciones BGI que no se utilizan en el paquete BATCH-GRAP:
InstallUser..., SetAspectRatio, GetMaxMode, GetModeRange, SetAllPalette, SetRGBPalette, SetActivePage, SetVisualPage, GraphDefaults.

5 LAS FUNCIONES DEL MANEJADOR DEL MOUSE

El uso de un MOUSE como dispositivo de entrada de datos es una labor de programación muy sencilla que consiste en depositar determinados valores en los registros del CPU, depositar en el registro AX el número del servicio deseado, invocar al driver del MOUSE ejecutando la interrupción INT 33H, y opcionalmente tomar los valores de regreso de los registros del CPU.

La forma de hacer lo anterior desde un lenguaje de alto nivel varía dependiendo del compilador. En el caso de Turbo-Pascal se cuenta con la función "intr", la cual usa como parámetro una variable de tipo "registers" ambos definidos en la "unit" Dos, por lo que el procedimiento es el siguiente:

1 - Indicar que se va a utilizar la "unit" Dos.
2 - Declarar una variable de tipo "registers".
3 - Almacenar en los registros los valores adecuados, dependiendo del servicio solicitado, y en el registro AX el número del servicio.
4 - Invocar al driver del MOUSE con la función "intr".
5 - Opcionalmente revisar los valores de regreso.

A continuación se muestra un ejemplo completo del procedimiento anterior:

```
program CHECA_MOUSE;
uses Dos;                  { 1-Indica que va a usar la "unit" DOS }
var
    Regs: registers;       { 2-Declara una variable de tipo "registers" }
begin
    Regs.AX := 0;          { 3-Almacena en AX el servicio deseado }
    intr($33,regs);        { 4-Invoca al driver del MOUSE }
    if Regs.AX = 0 then    { 5-Revisa el valor final de AX }
        writeln("EL MOUSE NO ESTA FUNCIONANDO")
    {noelse}
    {endif};
end.
```

En el caso del compilador Turbo-C también se cuenta con una función "intr" prácticamente idéntica a la de Turbo-Pascal, con la única excepción de que su parámetro es de tipo "struct REGPACK", por lo que el procedimiento para utilizar el MOUSE es el mismo. En seguida se muestra un ejemplo completo:

```
#include <dos.h>             /* 1-Indica incluír el "header" DOS */
#include <stdio.h>
struct REGPACK               /* 2-Declara la variable adecuada */
    Regs;
main()
    {
    Regs.r_ax = 0;           /* 3-Almacena en AX el servicio deseado */
    intr(0x33,&regs);        /* 4-Invoca al driver del MOUSE */
    if (Regs.r_ax == 0)      /* 5-Revisa el valor final de AX */
        puts("EL MOUSE NO ESTA FUNCIONANDO");
    /*noelse*/
    /*endif*/
    }
/*endmain*/
```

En el caso del compilador Clipper la biblioteca GRAPPER.LIB proporciona la función MOUSEFUNC la cual permite colocar directamente en sus parámetros los valores que se depositarán en los registros del CPU, y la función se encargará de invocar al driver del MOUSE, depositar los valores de regreso en los parámetros que fueron precedidos por el caracter @ y proporcionar de regreso el valor final del registro AX de acuerdo con la siguiente forma general:

```
AX_final = MOUSEFUNC(AX,[@]BX,[@]CX,ES:[@]DX,SI,DI)
```

Todos los parámetros deben ser de tipo entero excepto el cuarto, el cual puede ser entero, en cuyo caso su valor se depositará en el registro DX, o bien de tipo string, en cuyo caso su dirección se depositará en el par de registros ES:DX. Si cualquier parámetro se omite se inicializará con un cero.

Utilizando esta función el procedimiento de uso del MOUSE es el siguiente:

1 - Invocar al driver del MOUSE con la función MOUSEFUNC colocando en los parámetros los valores adecuados, dependiendo del servicio solicitado y en el primer parámetro el número del servicio.
2 - Opcionalmente revisar el valor final de la función como registro AX, o el valor final depositado en los parámetros correspondientes.

A continuación se muestra un ejemplo completo del procedimiento anterior:

```
IF MOUSEFUNC(0) = 0
   ? "EL MOUSE NO ESTA FUNCIONANDO"
*NOELSE
ENDIF
```

En seguida se describen todas las funciones que deben proporcionar los drivers de MOUSE compatibles con el driver del MOUSE de Microsoft versión 6. Esta descripción se proporciona para Clipper por ser la más corta y sencilla; sin embargo, la traducción del uso de los registros en Turbo-Pascal o Turbo-C es inmediata. Para una mayor descripción de estas funciones consulte el manual técnico del MOUSE de Microsoft (Microsoft Mouse Programmer's Guide).

```
BUTTONS = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
STATUS = MOUSEFUNC(0,@BUTTONS)
```

Inicializa el driver del MOUSE y regresa su estatus, si es -1 el MOUSE opera correctamente y BUTTONS indica cuántos botones tiene (2 ó 3), si el estatus es 0 el MOUSE no trabaja. Después de inicializar al MOUSE su cursor gráfico (MOUSE-POINTER) estará oculto (no aparece en la pantalla).

```
MOUSEFUNC(1)
```

Muestra el MOUSE-POINTER en la pantalla y cancela cualquier zona de ocultación automática definida con la función 16.

```
MOUSEFUNC(2)
```

Oculta el MOUSE-POINTER. Si éste se oculta varias veces será necesario ejecutar la función 1 el mismo número de veces para volverlo a mostrar.

```
BUTTON_STAT= 0   && CUALQUIER VALOR ENTERO DE INICIALIZACION
X = 0
Y = 0
MOUSEFUNC(3,@BUTTON_STAT,@X,@Y)
```

Obtiene el estatus de los botones y la posición actual del MOUSE-POINTER. El valor de cada botón es: izquierdo=1, derecho=2 y central=4; si se oprimen simultáneamente varios botones se sumarán sus valores.

```
MOUSEFUNC(4,0,X,Y)
```

Mueve el MOUSE-POINTER a la posición indicada.

```
BUTTON = NUMERO DE BOTON (0=izquierdo, 1=derecho, 2=central)
X = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
Y = 0
BUTTON_STAT = MOUSEFUNC(5,@BUTTON,@X,@Y)
```

Obtiene el número de veces que se ha oprimido un cierto botón desde la invocación anterior a esta función. Al entrar BUTTON indica el botón por revisar de acuerdo con lo siguiente: 0=izquierdo, 1=derecho y 2=central; al salir la función regresará el estatus de los botones (ver la función 3), BUTTON tendrá el número de veces que se ha oprimido el botón indicado, y X e Y tendrán la posición de la última presión del botón.

```
BUTTON = NUMERO DE BOTON (0=izquierdo, 1=derecho, 2=central)
X = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
Y = 0
BUTTON_STAT = MOUSEFUNC(6,@BUTTON,@X,@Y)
```

Obtiene el número de veces que se ha liberado un cierto botón. Esta función es el complemento de la función anterior pero considerando ahora la liberación de botones en lugar de las presiones.

```
MOUSEFUNC(7,0,X_MINIMUM,X_MAXIMUM)
```

Fija los límites horizontales en los que se podrá mover el MOUSE-POINTER.

```
MOUSEFUNC(8,0,Y_MINIMUM,Y_MAXIMUM)
```

Fija los límites verticales en los que se podrá mover el MOUSE-POINTER.

```
MOUSEFUNC(9,X_POINT,Y_POINT,"MOUSE POINTER IMAGE: 32 WORDS")
```

Cambia la forma del MOUSE-POINTER gráfico. Para una mayor descripción de esta función consulte el manual técnico del MOUSE de Microsoft.

MOUSEFUNC(10,TYPE,AND_MASK_OR_START_LINE,XOR_MASK_OR_END_LINE)
Cambia el MOUSE-POINTER de texto. Para una mayor descripción de esta función consulte el manual técnico del MOUSE de Microsoft.

```
DELTA_X = 0   && CUALQUIER VALOR ENTERO DE INICIALIZACION
DELTA_Y = 0
MOUSEFUNC(11,0,@DELTA_X,@DELTA_Y)
```

Obtiene el desplazamiento del MOUSE en MICKEYS desde la invocación anterior a esta función. La unidad de movimiento del MOUSE se llama MICKEY y puede medir 1/100 de pulgada (en modelos anteriores), 1/200 de pulgada (el estándar actual) o 1/320 de pulgada (en alta resolución).

La función 12 del MOUSE sólo puede utilizarse desde ensamblador.

MOUSEFUNC(13)
Activa la emulación de pluma óptica (light pen). La condición de "pluma abajo" se simula al oprimir ambos botones del MOUSE simultáneamente.

MOUSEFUNC(14)
Desactiva la emulación de pluma óptica.

MOUSEFUNC(15,0,MICKEYS_X,MICKEYS_Y)
Fija el número de MICKEYS necesarios para avanzar 8 pixels al MOUSE-POINTER (sensibilidad del MOUSE). El default es X=8 e Y=16.

MOUSEFUNC(16,0,LEFT,TOP,RIGHT,BOTTOM)
Define una zona de ocultación automática para el MOUSE-POINTER; cuando penetre en esta zona, automáticamente se ocultará (ver la función 1).

Las funciones 17 y 18 del MOUSE no están definidas.

MOUSEFUNC(19,0,0,MICKEYS_PER_SECOND)
Fija una velocidad a partir de la cual se duplicará el movimiento del MOUSE-POINTER (mas velocidad implica mas movimiento). El default es 64, para cancelar esta característica fije un valor muy alto (p.e. 10000).

La función 20 del MOUSE sólo puede utilizarse desde ensamblador.

```
BUFFER_SIZE = 0  && CUALQUIER VALOR ENTERO DE INICIALIZACION
MOUSEFUNC(21,@BUFFER_SIZE)
```

Obtiene el tamaño del buffer de almacenamiento del estatus del MOUSE.

```
BUFFER = SPACE(BUFFER_SIZE)
MOUSEFUNC(22,0,0,@BUFFER)
```

Almacena el estatus actual del MOUSE.

MOUSEFUNC(23,0,0,@BUFFER)
Recupera un estatus del MOUSE previamente almacenado con la función 22.

Las funciones 24 y 25 del MOUSE sólo pueden utilizarse desde ensamblador.

MOUSEFUNC(26,MICKEYS_X,MICKEYS_Y,MICKEYS_PER_SECOND)
Fija la sensibilidad (función 15) y velocidad de duplicación (función 19) del MOUSE pero expresadas en porcentajes (de 0 a 100), el default es 50.

```
MICKEYS_X = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
MICKEYS_Y = 0
MICKEYS_PER_SECOND = 0
MOUSEFUNC(27,@MICKEYS_X,@MICKEYS_Y,@MICKEYS_PER_SECOND)
```
Obtiene los valores fijados con la función anterior.

MOUSEFUNC(28,INTERRUPT_RATE)
Fija la frecuencia de interrupción del MOUSE InPort. Para una descripción de esta función consulte el manual técnico del MOUSE de Microsoft.

MOUSEFUNC(29,PAGE)
Fija la página de video en la que se mostrará el MOUSE-POINTER.

```
PAGE = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
MOUSEFUNC(30,@PAGE)
```
Obtiene la página de video fijada con la función anterior.

Las funciones 31 y 32 del MOUSE sólo pueden utilizarse desde ensamblador.

```
BUTTONS = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
STATUS = MOUSEFUNC(33,@BUTTONS)
```
Reinicializa el driver del MOUSE y regresa su estatus. Esta función es similar a la función 0 excepto que no inicializa el hardware.

MOUSEFUNC(34,LANGUAGE)
Fija el lenguaje de los mensajes del driver del MOUSE de acuerdo con: 0=Inglés, 1=Francés, 2=Holandés, 3=Alemán, 4=Sueco, 5=Finlandés, 6=Español, 7=Portugués y 8=Italiano (sólo en versiones multilingües).

```
LANGUAGE = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
MOUSEFUNC(35,@LANGUAGE)
```
Obtiene el lenguaje fijado con la función anterior.

```
VERSION = 0    && CUALQUIER VALOR ENTERO DE INICIALIZACION
TYPE = 0
MOUSEFUNC(36,@VERSION,@TYPE)
```
Obtiene el número de versión del driver y tipo del MOUSE. Para una mayor descripción de esta función consulte el manual del MOUSE de Microsoft.

6 APENDICES

6.1 DESCRIPCION BASICA DE LA "TEORIA DEL COLOR"

La siguiente descripción fué tomada del libro "High-Performance Graphics in C, Animation and Simulation" de Lee Adams, ed. Windcrest. En muchos casos se conservaron las palabras originales en inglés debido a que escoger una cierta palabra en español podría cambiar el significado del texto. Todas las palabras que no se tradujeron están encerradas entre comillas, y cuando fué posible se incluyó en seguida una traducción aproximada encerrada entre paréntesis.

TERMINOLOGIA DEL COLOR.

Cuando los profesionales hablan del color, utilizan una terminología específica. Los colores cromáticos son aquellos que tienen un determinado "hue" (color, matiz): rojo, amarillo, verde, etc; los colores acromáticos son neutrales: blanco, gris y negro; estos colores no tienen "hue". Es importante entender la diferencia entre "hue", "value" (valor) y "chroma" (este término se emplea igual en español, o se usa el de "intensidad"). "Hue" (color, matiz) es lo que distingue un color cromático de otro. Su uso más efectivo es en las Ruedas de Colores en las que son utilizados para crear otras formas de color con el fin de producir efectos emocionales, efectos sicológicos o impacto visual. Existen dos clases de "hues", "warm hues": rojo, amarillo, naranja, etc., y "cool hues": verdes y azules. "Value" se refiere a la brillantez u obscuridad aparentes de un color. "Chroma" se refiere a la pureza relativa de un color. Por ejemplo, el naranja tiene fuerte "chroma", mientras que el canela la tiene pobre. La "chroma" también se llama intensidad o saturación.

"SHADES, TONES and TINTS".

Las "shades" (sombras) se forman mezclando colores puros con negro. El café, el marrón y el oliva son "shades". Los "tones" (tonos) se forman mezclando colores puros con gris. El canela y el beige son "tones". Las "tints" (tintas) se forman mezclando colores puros con blanco. El rosa y el durazno son "tints" Los "shades", "tones" y "tints" creados del mismo "warm hue" son marcadamente diferentes. Por ejemplo, el rosa ("tint") es definitivamente distinto del rojo (el "hue" original); el café ("shade") no se parece al naranja (el "hue" original). Por otro lado, los "shades", "tones" y "tints" creados del mismo "cool hue" tienden a permanecer esencialmente el mismo tipo de color. Un "shade" verde y una "tint" verde se parecen mucho al "hue" verde original.

CIRCULOS DE COLOR.

La rueda de colores de la derecha se deriva del círculo de color estándar de la izquierda. Un círculo de color se forma con los 4 "hues" básicos que puede reconocer el ojo humano: rojo, amarillo, verde y azul. Usando la física de la luz -que es el mismo mecanismo utilizado por el monitor de la computadora- los 3 "hues" básicos rojo/verde/azul pueden ser entremezclados para formar todos los otros colores puros mostrados en la rueda de colores. Esta es la razón por la cual los monitores son llamados "RGB", ya que utilizan puntos de fósforo "Red", "Green" y "Blue" para crear el rango completo de colores.

Amarillo

Naranja Verde

Rojo Azul

Morado

Amarillo

Amarillo-naranja Amarillo-verde

Naranja Verde

Rojo Azul-verde

Magenta Morado Violeta Azul Turquesa

ARMONIAS DE COLOR.

Ciertas combinaciones de colores hechas de una forma determinada dentro del círculo de color producen efectos sicológicos y emocionales característicos. Estas combinaciones son llamadas armonías y se muestran enseguida.

Armonía análoga.

Una armonía análoga es una combinación de "hues" que están adyacentes en el círculo de color, por ejemplo: amarillo/naranja/rojo. Las armonías análogas producen los efectos sicológicos que corresponden a los "warm hues" o "cool hues" utilizados en la armonía. Una combinación análoga de colores que incluya azul, rojo o verde es percibida como directa y franca. Una sensación de algo mas discreto y elegante se consigue con una armonía análoga basada en naranja, violeta o amarillo-verde, etcétera.

Armonías complementarias.

Las armonías complementarias son de dos tipos: armonía directa y armonía de complemento dividido. Una armonía directa es una combinación de colores que están directamente opuestos en el círculo de color, como sería la combinación de rojo y verde. Las armonías directas producen un fuerte impacto visual.
Las armonías de complemento dividido se forman combinando colores que están colocados en un triángulo dentro del círculo de color, como sería el caso de la combinación rojo/amarillo/azul; estas combinaciones también se llaman armonías triádicas. La combinación magenta/amarillo/turquesa se percibe como primitiva; la combinación rojo/amarillo-verde/azul se percibe como ostentosa; la combinación morado/amarillo-naranja/azul-verde se percibe como apropiada; y la combinación naranja/verde/violeta se percibe como delicada.

ESCALAS DE COLOR.

Un sólo "hue" puede ser mezclado con diferentes cantidades de blanco puro y negro puro para producir un número ilimitado de "shades", "tones" y "tints". El esquema mostrado a continuación muestra la misma metodología seguida por los fabricantes de pinturas para controlar en forma precisa los colores de sus productos. Las mezclas que varían desde el "hue" puro hasta el blanco puro forman la escala de "tints", las mezclas que van del "hue" puro al negro puro forman la escala de "shades". Las mezclas que van del blanco puro al negro puro forman la escala de gris, las mezclas formadas por el "hue" puro en combinación con la escala de gris se llaman "tones". En la siguiente figura se representan

las fórmulas precisas de mezcla utilizadas por este enfoque científico para la creación de "shades", "tones" y "tints" mediante el porcentaje que tiene cada una de ellas del "hue"-puro/blanco/negro.

		"TINTS"		BLANCO	
		DE		0/100/0	
	ESCALA				E
			10/90/0		S
					C
		25/75/0		0/90/10	A
			20/75/5		L
	50/50/0	45/50/5			A
"HUE" PURO					
100/0/0	75/20/5	60/30/10	40/45/15	0/75/25	D
					E
	90/0/10	60/20/20			
			40/30/30		G
		75/0/25		0/50/50	R
					I
			50/0/50		S
	ESCALA				
		DE		0/0/100	
		"SHADES"		NEGRO	

ESCALAS.

El ojo humano puede distinguir nueve diferentes tonos de gris entre el blanco puro y el negro puro; la escala correspondiente se muestra a continuación:

	NEGRO	1	2	3	4	5	6	7	8	9	BLANCO
NEGRO:	100%	85%	65%	50%	35%	25%	15%	10%	5%	2%	0%
BLANCO:	0%	15%	35%	50%	65%	75%	85%	90%	95%	98%	100%

Las capacidades del lenguaje Turbo-C hacen posible al programador el simular esta escala de gris en el video definiendo diversos patrones de relleno con diferentes porcentajes de pixels blancos sobre un fondo de color negro. Aún las escalas de gris de otros "hues" pueden mostrarse en la pantalla, incluídos los azules, verdes, "cyans", rojos, magentas y cafés de la paleta estándar de 16 colores de las tarjetas EGA y VGA. Esto significa que los "hues" estándar de la paleta de 16 colores pueden ser expandidos a una paleta de alrededor de 150 "shades", todos los cuales pueden ser mostrados simultáneamente en la pantalla. Si se mezclaran diferentes "hues" (diferentes porcentajes de azul y verde, por ejemplo) alrededor de 2,400 diferentes mezclas de colores podrían crearse y mostrarse simultáneamente en cualquier modo de video de 16 colores.

SICOLOGIA AVANZADA DEL COLOR.

Usted puede sorprenderse al saber que cualquier "hue" puede ser manipulado en la pantalla de la computadora para producir una amplia variedad de efectos sicológicos. La técnica utilizada es el mismo enfoque científico usado por los artistas que han estudiado la teoría del color. Por ejemplo, el mismo "hue" rojo puede ser manipulado para producir los siguientes efectos visales:

* Empañado, como neblina brumosa.
* Transparente, como vidrio teñido.
* Luminoso, como un semáforo.
* Lustroso, como la seda.
* Metálico, como lentejuelas.
* Iridiscente, como una perla.

Ley del Tamaño del Campo.

Es la Ley del Tamaño del Campo lo que hace posible estas transformaciones de percepción. Para decirlo en forma simple, el fondo contra el cual se muestra el "hue" rojo modifica la percepción que tiene el ojo de la calidad del "hue". En todos los ejemplos mostrados anteriormente no es el "hue" rojo el que cambia en la pantalla de video, sino el fondo contra el cual se muestra.
Un campo obscuro (formado por "shades") simula una baja iluminación y produce efectos lustrosos. Un campo claro (formado por "tints") simula una iluminación brillante y produce un efecto cromático brumoso. Un campo grisáceo (formado por "tones") simula niebla y distancia, y produce efectos iridiscentes. Un par de ejemplos demostrarán cómo trabaja esta técnica avanzada.

Efectos lustrosos.

Por ejemplo, suponga que desea que el "hue" rojo aparezca lustroso, como el brillo de la seda fina. Puede lograr esto manteniendo el fondo de las gráficas formado por "shades" ("hues" puros mezclados con porcentajes variables de negro puro). Ahora, cualquier "hue" puro (el rojo) que se muestre sobre el fondo aparecerá excepcionalmente brillante en comparación.

Efectos iridiscentes.

Suponga que desea que el "hue" rojo aparezca iridiscente, como el brillo inherente de un ópalo o una perla. En la naturaleza, esta iridiscencia es realmente causada por la difracción de los rayos de luz. Manteniendo el fondo de la pantalla de video en tonos grisáceos (formados por "tones"), el ojo es engañado y percibe brumosidad y un "chroma" uniformemente reducido. El "hue" rojo puro aparecerá entonces brillando de una manera iridiscente.

Como puede verse claramente, el sorprendente potencial de las tarjetas gráficas de video EGA y VGA apenas ha sido tocado por el software existente. Estudiando los conceptos anteriores, y definiendo sus propios patrones de relleno para la creación de "shades", usted puede comenzar a explorar el excitante campo de la teoría del color.

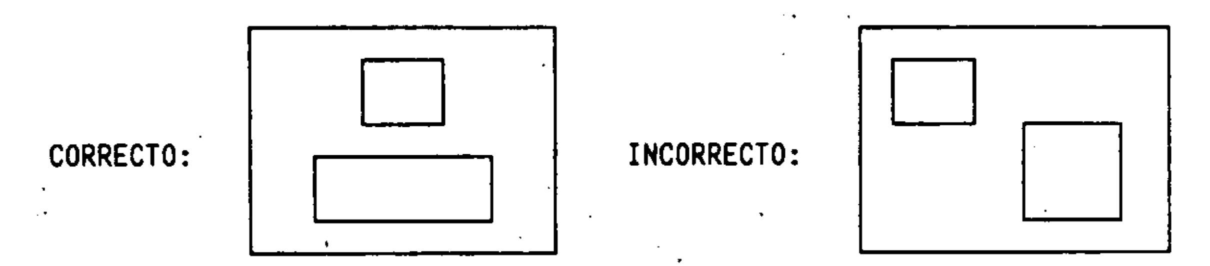
6.2 RECOMENDACIONES BASICAS SOBRE LA DISPOSICION

La disposición que guardan entre sí todos los elementos de una presentación gráfica ("layout") es muy importante, al grado que existen teorías y extensos estudios al respecto. Sin entrar en mayores detalles, podemos beneficiarnos de esas teorías extrayendo la parte central de sus ideas y aplicándolas mediante reglas y recomendaciones muy sencillas de seguir. A continuación se presentan una serie de consejos extraídos del mismo libro mencionado en el apéndice anterior; estas recomendaciones se refieren a la posición, color y tamaño que deben tener los distintos elementos que componen una presentación gráfica.

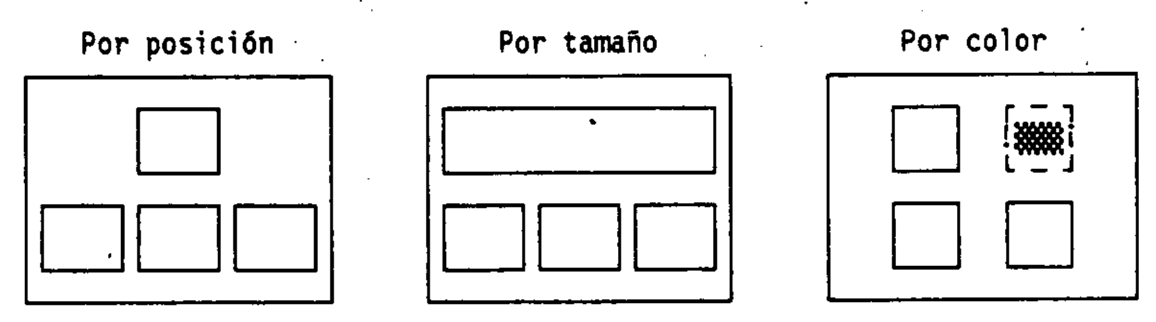
El primer consejo consiste en preparar previamente sobre una hoja de papel un esbozo general ("sketch") de la presentación gráfica que incluya el sistema de coordenadas que tendrá el área de graficación. En este esbozo es muy sencillo hacer cambios hasta conseguir el aspecto deseado; sólo se debe comenzar a hacer la presentación gráfica en sí (el programa de computadora) cuando ya se cuente con una idea precisa de todas las partes que formarán la presentación.

El centro visual de una zona similar a la pantalla de la computadoa no se encuentra en el centro de la misma, sino un poco hacia arriba de la mitad. El centro visual debe contener la parte más importante de una presentación.

Si una presentación contiene varios elementos de interés se debe dividir la pantalla en varias zonas, procurando que esta división tenga un balance visual que dé la idea de orden y planeación, por ejemplo:



Siempre se debe procurar darle un mayor énfasis a una de las zonas, ya que el tener una pantalla dividida en varias zonas diferentes pero todas con el mismo aspecto dá una sensación de ambigüedad que frustra al auditorio. Hay varias formas de darle énfasis a una zona en particular, por ejemplo:



También se debe hacer un uso adecuado del color; para una mayor descripción de este punto revise la Teoría del Color en el apéndice anterior.

Una regla general es la de evitar ambigüedades, ya que éstas frustran a la persona que ve la pantalla y la hacen perder interés. Si se desea mostrar un círculo o una elipse, éstos debe tener formas bien definidas y no verse como un círculo ligeramente achatado que nadie pueda decidir si es una elipse o no; esto mismo se aplica al mostrado de cuadrados y rectángulos.

En forma similar, si dos figuras aparecen adyacentes no debe haber ninguna duda de cuáles líneas forman a cada una de ellas. Si por lo menos una línea forma parte de las dos figuras entonces hay una ambigüedad, que debe evitarse.

Los encabezados, títulos, subtítulos, etc. deben aparecer centrados, ya sea en toda la pantalla, o bien, dentro de la zona a la cual pertenecen.

Si se tiene un texto principal éste debe estar justificado a la izquierda pero no a la derecha ("ragged"); el texto que se justifica a la derecha tiene un mejor balance visual como un todo, pero es más difícil de leer.

Para resaltar un texto se debe usar un contraste de color, sin embargo, si se desea que un texto tenga mucho impacto entonces se debe aumentar su tamaño; la posición de un texto afecta muy poco el impacto que éste tiene.

Cuando se combina texto con gráficas los encabezados deben ir en la parte superior y centrados. Un texto se podrá poner en el centro de la pantalla sólo si las gráficas que lo acompañan no son lo más importante. Los subtítulos o descripciones de las gráficas deben ir en la parte inferior y centrados.

Los investigadores han descubierto que la mayoría de las personas revisan los elementos de una nueva presentación en el siguiente orden: 1- las gráficas, 2- los subtítulos, 3- los encabezados y 4- las descripciones; conviene tomar en cuenta este orden al preparar ciertas presentaciones muy especiales.

Finalmente, el último consejo es una repetición del primero: siempre haga un prototipo; el tiempo invertido en planear y preparar un "sketch" con cuidado sobre hojas de papel se ganará con creces cuando éste se pase a programar.

*